Método de evaluación de desempeño de algoritmos de planificación de trayectorias en tareas de navegación con objetivos dirigido a personas invidentes

Paula Andrea Mosquera Ortega

Asesor

Andrés Díaz Toro

Universidad Nacional Abierta y a Distancia UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería ECBTI

Maestría en Gestión de Tecnología de Información

2023

Nota de aceptación

Firma del presidente del Jurado

Firma del Jurado

Firma del Jurado

Pasto, 2023

Dedicatoria

A mi madre, mi esposo, mis hijas y mis hermanos dedico las ganas de culminar con éxito el trabajo que hace 3 años comencé, a mi padre que aún lo siento cerca, sin lugar a dudas el sería el más orgullo de todos. A Dios que acompaña cada paso de mi vida gracias, por ponerme a mi lado personas grandiosas e incondicionales, por ellos siempre fuerte.

Agradecimientos

A mi director y asesor de tesis de grado, el Doctor Andrés Díaz Toro, por su guía y direccionamiento para llegar al objetivo, por su tiempo, paciencia, por su dedicación y empeño para culminar con éxito este proyecto de investigación.

Mil gracias a Santiago Insuasty por confiar en que sí era posible, a William Insuasty mi compañero de vida, por estar siempre apoyando este proceso.

A mis compañeros de trabajo UNAD CCAV Pasto, por su apoyo y buenas energías.

Resumen

En el mundo existen algunos estudios sobre la tecnología de visión por computador para vehículos autónomos, cómo: transporte público autónomo, camiones inteligentes, vehículos sin conductor, entre otros, pero hay pocos trabajos que busquen asistir en navegación a personas invidentes, quienes no pueden transitar fácilmente por entornos desconocidos. Acorde a ello, para poder implementar las tecnologías que asistan en navegación a estas personas, se necesita identificar los mejores algoritmos que detecten y evadan obstáculos en entornos dinámicos, que sean de bajo costo, que localicen objetos de interés, y que guíen al usuario a dichos objetos a través de trayectorias óptimas. En contexto con un proyecto que se viene desarrollando cómo estancia postdoctoral denominado "Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real," al cuál se ha llamado AssistnavBP, en su segunda fase se desea detectar objetos de interés y generar trayectorias optimas a través de un algoritmo de planificación de trayectorias. Ya que las tecnologías para detectar objetos y generar trayectorias optimas hacia un objetivo requieren de equipos de alto desempeño y de tamaño pequeño que hace algunos años no existían, hoy ya se cuenta con tarjetas gráficas que garantizarían la eficiencia del equipo. En este trabajo de grado para optar el título de Magister en Gestión de Tecnología de Información, se busca evaluar algoritmos que permitan detectar trayectorias eficientes con facilidad, creando un método que apruebe integrar hardware y software escogiendo y observando el desempeño de algunos algoritmos de búsqueda, implementarlos y proponer el que mejor se adapte a las condiciones del proyecto inicialmente planteado. El método desarrollado tiene como objetivo guiar a un investigador interesado en implementar un algoritmo de planificación de trayectorias, en la evaluación, selección e implementación del

algoritmo más apropiado, de forma sistemática y eficiente. Finalmente se busca ayudar a los invidentes en navegación en entornos desconocidos, y así mejorar su calidad de vida.

Palabras clave: algoritmos, desempeño, evaluación, gestión, invidentes, TIC, método.

Abstract

In the world there are some studies on computer vision technology for autonomous vehicles, such as: autonomous public transport, intelligent trucks, driverless vehicles, among others, but there are few works that seek to assist in navigation for blind people, who cannot easily navigate through unfamiliar environments. Accordingly, in order to implement technologies that assist in navigation to these people, it is necessary to identify the best algorithms that detect and avoid obstacles in dynamic environments, are low cost, locate objects of interest, and guide the user to these objects through optimal trajectories. In context with a project that is being developed as a Post Doctoral stay called " Portable system to assist blind people in purposeful navigation at real time " which has been called AssistnavBP, in its second phase, it is desired to detect objects of interest and generate optimal trajectories through a trajectory planning algorithm. Since the technologies to detect objects and generate optimal trajectories to a target require high performance equipment and small size that did not exist until few years ago, today there are already graphic cards that would ensure the efficiency of the equipment. In this work for the title of Master in Information Technology Management, we seek to evaluate algorithms that allow to detect efficient trajectories easily, creating a method that approves to integrate hardware and software by choosing and observing the performance of some search algorithms, implement them and propose the one that best suits the conditions of the project initially proposed. The developed method aims to guide a researcher interested in implementing a path planning algorithm, in the evaluation, selection and implementation of the most appropriate algorithm, in a systematic and efficient way. Finally, it seeks to help blind people in navigation in unfamiliar environments, and thus improve their quality of life.

Keywords: algorithms, performance, evaluation, management, blind, ICT, method.

Tabla de Contenido

Tabla de Figuras

Lista de Tablas

Lista de Apéndice

Introducción

La vista, uno de nuestros sentidos más predominantes, desempeña un papel esencial en todas las facetas y etapas de nuestra existencia. A menudo, damos por sentado esta capacidad, pero su ausencia dificultaría nuestras habilidades para aprender, desplazarnos, leer, participar en la educación y mantener empleos. La discapacidad visual surge cuando alguna afección ocular afecta el sistema visual y sus funciones relacionadas con la vista. En el transcurso de nuestras vidas, la mayoría de nosotros experimentará al menos una condición ocular que requerirá atención médica, siempre y cuando vivamos lo suficiente. (Organización Mundial de la Salud, 2023).

La problemática de los invidentes en el mundo da muchas opciones para ayudar a la creación de herramientas y facilitar su vida a pesar su la discapacidad. No solamente se ha trabajado en la movilidad de personas con discapacidad visual, también se encuentran investigaciones que a partir de la planificación de trayectorias buscan mejorar la calidad de vida de personas con enfermedades como alzhéimer, demencia, enfermedades de movilidad, entre otras. Las nuevas tecnologías como; la inteligencia artificial, la robótica, los asistentes de voz, análisis de datos, vehículos autónomos son apenas algunos ejemplos de las herramientas para la creación de dispositivos que ayuden a personas con problemas de movilidad. La planificación de trayectorias permite encontrar una ruta que llegue a un objetivo evitando obstáculos y así asegurando el éxito en el camino. Este trabajo busca ser una guía para apoyar a cualquier investigador que trabaje con diferentes disciplinas, donde se trate de encontrar la mejor ruta en un desplazamiento, y entregar una metodología que apoye en el paso a paso de cómo estructurar la secuencia para analizar que algoritmo de planificación de trayectorias es el más eficiente, la utilidad puede ser muy grande.

Problema de Investigación

En este capítulo se tratará el planteamiento del problema, la justificación, el objetivo general y los objetivos específicos.

Pregunta de Investigación

¿En qué medida la implementación de un método de evaluación de desempeño de algoritmos de planificación de trayectorias facilitará la selección del algoritmo más adecuado para asistir a personas invidentes utilizando criterios de optimización de tiempo y/o espacio?

La planificación es una rama de los algoritmos. Planificar es proponer diferentes opciones que nos permiten predecir los resultados y que a su vez puede entregar la mejor posibilidad en función de bajos esfuerzos.

Definir un camino que nos lleve de un punto A a un punto B y que ese camino sea optimo es la definición general de un algoritmo de planificación. Los algoritmos de planificación permiten realizar tareas de alto nivel con el fin de evitar esfuerzos innecesarios que requieren mucho más trabajo. Se pueden utilizar en infinidad de disciplinas que requieran la solución de un problema (Kelly, 2017).

El proyecto AssistnavBP parte inicialmente del problema: la persona invidente tiene dificultades para moverse en entornos interiores o exteriores no familiares y las herramientas que existen para que la persona invidente pueda movilizarse no son las adecuadas. Una persona invidente no tiene acceso a un asistente tecnológico para llegar al objeto de interés (Díaz et al., 2019).

Para detectar obstáculos se utiliza:

Bastón blanco: necesita detectar los obstáculos a corta distancia, lo hace por contacto que resulta ser peligroso, éste no entrega información del entorno en el que se encuentra la persona,

por ejemplo, qué tipo de objetos se encuentran en un lugar cerrado, tampoco la trayectoria que se debe seguir para llegar a un objetivo.

El perro guía: es una herramienta muy costosa, es necesario darle muy buen trato y necesita ser mantenido, además tiene una vida útil muy limitada.

También existen herramientas que son basadas en objetos electrónicos de ultrasonido y en visión, pero hay muy pocos sistemas de navegación con objetivo para invidentes que trabajen técnicas de visión por computador.

Inicialmente en el proyecto presentado por Díaz et al., (2019). "Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real", proponen dos etapas.

Crear una grilla de ocupación 2D que identifica qué celdas están libres y cuáles están ocupadas. El usuario puede explorar el entorno sin peligro de toparse con objetos. Esta etapa está concluida.

En la segunda etapa se quiere detectar objetos y generar trayectorias optimas hacia el objetivo.

En la primera etapa se tiene un sistema que sólo detecta obstáculos, y se quiere aumentar la funcionalidad incluyendo un módulo de planificación de trayectorias para llegar a un objeto de interés. Es por lo que frente al mejoramiento de la propuesta se pretende optimizar con un algoritmo de búsqueda. Para la segunda etapa se requiere de dispositivos que trabajen con tarjetas gráficas y equipos de cómputo que garanticen alta eficiencia y poco tamaño. Sin embargo, la atención se ha puesto frente a la navegación autónoma de vehículos (Gianibelli et al., 2018).

Se han desarrollado técnicas de visión por computador (Ruiz et al., 2012), detección de objetos y entendimiento del entorno sobre este tipo de aplicaciones, según Barba et al., (2017),

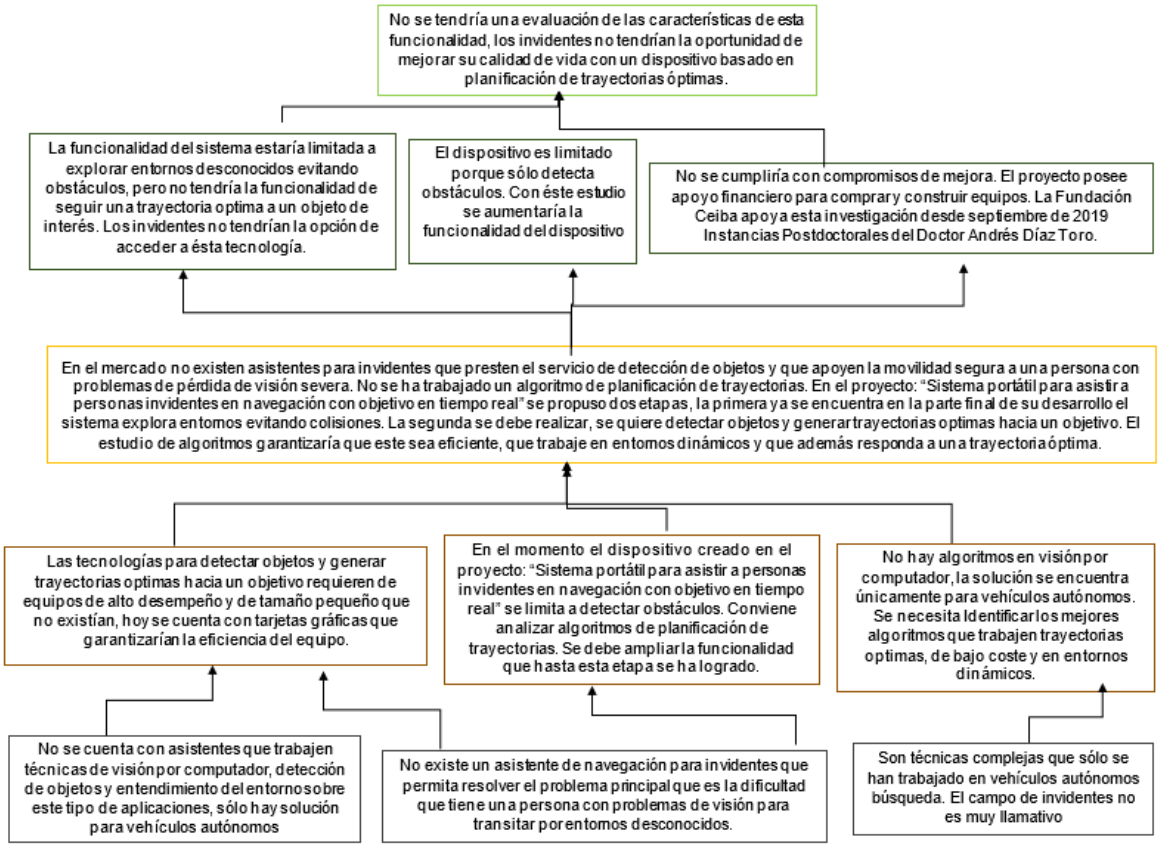
pero no para asistir a invidentes. En este trabajo se propone manejar algunas de estas técnicas utilizadas en vehículos autónomos y aplicarlas a asistentes para invidentes.

El sistema AssistNavBP consta de seis módulos para segmentar el piso, construir una cuadrícula de ocupación 2D, evitar obstáculos, detección y orientación de objetos, la planificación del camino y para la retroalimentación háptica. La propiedad principal del algoritmo de planificación de trayectorias es que sea robusto a objetos en movimiento o personas que interfieren con la trayectoria estimada, logrando adaptar la trayectoria en tiempo real a las nuevas condiciones del entorno. (Díaz et al., 2019), abriendo así un espacio de investigación muy poderoso y extenso en el campo de la Inteligencia Artificial IA, utilizando los algoritmos que se han creado para diferentes proyectos de robótica.

En este proyecto se busca evaluar el desempeño de algoritmos de planificación de trayectorias, teniendo en cuenta que deben ser eficiente, que trabaje en entornos dinámicos y que además garantice una trayectoria óptima. A continuación, se presenta el árbol de problemas donde se identifica la situación general, apoyo para determinar los objetivos y estrategias para poder cumplirlos.

Figura 1

Árbol de Problemas



Fuente. Elaboración propia del documento (2023)

Además, dado que no existe, se busca mediante una serie de pasos crear un método que sirva para guiar a una persona interesada en implementar un algoritmo de planificación de trayectorias, en cualquier aplicación, en la evaluación, selección e implementación del algoritmo más apropiado, de forma sistemática y eficiente.

Justificación

Esta investigación tiene como punto de partida los resultados del proyecto: “Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real” (Díaz et al., 2019).

En 2018 el proyecto AssistNavBP participó en los Premios de Google de Investigación de América Latina, Google LARA, siendo ganador y haciéndose acreedor del apoyo financiero para comprar y construir equipos que fortalezcan el proyecto de asistencia a invidentes. Gracias a esto y a la fundación CEIBA que apoya la investigación se ha mantenido la motivación para cumplir los objetivos. El trabajo que se debe realizar es largo y requiere de muchos más estudios para garantizar la eficiencia del dispositivo en la segunda etapa del proyecto.

El objetivo inicial era crear un dispositivo que apoyara el movimiento seguro, en un entorno desconocido, de una persona con problemas de visión. Luego de lograr este objetivo se busca analizar propuestas donde se utilicen algoritmos de planificación de trayectorias, y así implementar el algoritmo más adecuado en el sistema y poder garantizar que las nuevas tecnologías lleguen a las personas que padecen cada día por su discapacidad. Esto se desea realizar mediante un método que muestre los pasos que harían que cualquier persona interesada en implementar un algoritmo de planificación de trayectorias, en cualquier aplicación, pueda hacerlo de forma más eficiente.

Dotar a la comunidad de invidentes con sistemas portátiles de navegación autónoma (Peralta y Urmendiz 2014), lograría una fuerte independencia que no se ha logrado jamás. Según la Organización Mundial de la Salud, aproximadamente 1.300 millones de personas vivían con algún tipo de discapacidad visual en 2018. De ellas, 36 millones de personas eran ciegas (Organización Mundial de la Salud, 2018, p.1), lo que nos hace suponer que la cantidad de

beneficiados por la generación de nuevas tecnologías en este sector es muy grande y hace que el estudio tienda a proyectarse al futuro, con la seguridad que resulte un trabajo de alto impacto para la sociedad en el mejoramiento de la calidad de vida de esta población.

Las últimas investigaciones de planificación de trayectorias se han centrado en el trabajo sobre vehículos autónomos, el generar trayectorias optimas, desplazamiento asistido para evitar obstáculos. Las investigaciones en tiempo real no son muy comunes y en entornos dinámicos se restringen aún más, (Patle et al., 2019). Es muy frecuente encontrarse con análisis sobre robots inteligentes que son capaces de movilizarse con muy pocas instrucciones (Bruemmer y Swinson. 2003). Éstos son pequeños ejemplos en los que se pueden utilizar algoritmos de planificación de trayectorias. Bajo este análisis no se observa que los estudios se centren en un sistema que asista a invidentes con tecnología de vehículos autónomos. Se evidencia que el campo de acción, si se realiza esta investigación, es muy extenso, existen muchas bases teóricas que argumentan la necesidad de trabajar la segunda fase del proyecto AssistnavBP, además existen muchos algoritmos que han basado su aplicación en entornos desconocidos que apoyarían nuestro estudio (Patle, et al., 2019).

El proyecto necesita un sistema que identifique un camino optimo que le permita una persona con problemas de visión desplazarse con facilidad en entornos dinámicos y alcanzar un objeto de interés. Se desea aportar haciendo la implementación de algoritmos en Matlab y así poder sugerir el algoritmo que mejor trabaje en ese entorno. A estas alturas es una realidad que es necesario seguir invirtiendo el tiempo para realizar búsquedas profundas en el tema.

Objetivos

Objetivo General

Desarrollar un método de evaluación de desempeño de algoritmos de planificación de trayectorias, para asistir a personas invidentes, utilizando criterios de optimización de tiempo y/o espacio aplicado al proyecto “Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real”.

Objetivos Específicos

Realizar un análisis diagnóstico situacional para la determinación de la problemática en los métodos, procedimientos y algoritmos utilizados actualmente para generar las trayectorias de las personas invidentes y las diversas metodologías que se usan en la evaluación mediante un análisis documental.

Diseñar un procedimiento para la evaluación de algoritmos de planificación de trayectorias para personas invidentes, utilizando criterios de optimización de tiempo y/o espacio.

Validar el método de evaluación de algoritmos en un caso práctico basado en el proyecto “Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real”

Presentar documento final mediante consolidación de informes. Difusión de resultados.

Marco de Referencia

En este capítulo se muestra los conceptos más relevantes utilizados para desarrollar el proyecto, estos son: método, planificación de trayectorias, discapacidad visual, tecnologías asistidas, algoritmos, desempeño de algoritmos, grafos, nodos camino, terrenos, entornos, heurística, ciclos, unidades de procesamiento (CPU) y (GPU).

Método

"Un método es considerado como el camino para obtener un fin de manera ordenada, desde un conjunto de reglas, es una manera ordenada y consecuente de elaborar procesos para llegar a un resultado" (Gordillo, 2007, p.120).

Puede definirse como camino a seguir, mediante una serie de operaciones y reglas fijadas de antemano, de manera voluntaria y reflexiva para alcanzar cierto fin. Cubre varias significaciones: en sentido filosófico general o global, conjunto de actividades intelectuales que, con prescindencia de contenidos específicos, establece procedimientos lógicos, formas de razonar, que hacen accesible la realidad a captar. Los métodos ayudan a una mejor utilización de los medios para acceder al conocimiento de la realidad, a fijar de antemano una manera de actuar racional y eficaz, a operar sobre la misma realidad y a evaluar los resultados de la acción (Gordillo, 2007).

La Planificación de Trayectorias

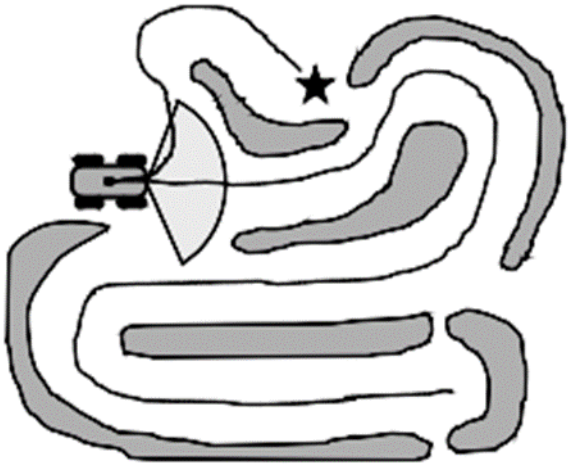
Se aplica en robótica, inteligencia artificial y teoría de control; que son temas generales e individuales y requieren para este análisis ser trabajados en conjunto. Espacios tan diversos como la biología computacional, la creación de prototipos virtuales en la fabricación, el diseño arquitectónico, la ingeniería aeroespacial y la geografía computacional, en la industria, la

medicina, y en particular en dispositivos que ayudarán a sobrellevar la pandemia, se aplican la planeación de trayectorias.

Particularmente en: Rompecabezas discretos, operaciones y programación, rompecabezas de planificación de movimiento, problemas de ensamblaje automotriz, sellado de grietas en el ensamblaje automotriz, navegación de robots móviles, humanos virtuales y robots humanoides, ubicación de automóviles y remolques, diseño de mejores medicamentos, aplicaciones aeroespaciales, etc. En la Figura 2 se observa un robot que pretende ir inicialmente en línea recta porque parece la mejor ruta, pero mirando hacia delante el mapa lo dirige hacia la izquierda, que es la ruta adecuada. Siendo éstos apenas unos ejemplos que motivan a la creación y utilización de algoritmos de planificación (Kelly, 2017).

Figura 2

Planeación de Trayectorias.



Fuente. Tomado de Kelly (2017, p.1).

Discapacidad Visual: Organización Mundial de la Salud

Según la - OMS, se estima que más de siete millones de personas quedan ciegas cada año.

75% de la ceguera en el mundo es evitable.

80% de las discapacidades visuales son evitables.

63% de las personas con baja visión y 82% de las personas ciegas tienen más de 50 años de edad.

De las seis regiones de la OMS, el sudeste de Asia y el Pacífico Occidental cuenta con el 73% de la discapacidad visual moderada a severa y representan el 58% del total de la ceguera en el mundo. “Hay 285 millones de personas con discapacidad visual en el mundo, 39.8 millones de ellos son ciegos” (Europa Press, 2014). La mayoría de las personas con discapacidad visual son mayores de 50 años, las mujeres tienen más riesgo de ceguera en todos los grupos etarios y en todas partes del mundo. 90% de las personas con discapacidades visuales viven en países en vías de desarrollo.

La discapacidad visual se refiere a personas con deficiencias funcionales del órgano de la visión y, de las estructuras y funciones asociadas, incluidos los párpados. Está determinada por los niveles de deterioro de la función visual, y que se establece tras la medición de la agudeza visual y del campo visual de cada uno de los ojos por separado.

La OMS subdivide la función visual en dos grupos según el tipo de visión: de lejos y de cerca. Los niveles del déficit visual de lejos son:

Leve – agudeza visual inferior a 6/12 (0,5) e igual o superior a 6/18 (0,3).

Moderada - agudeza visual inferior a 6/18 (0,3) e igual o superior a 6/60 (0,1).

Grave - agudeza visual inferior a 6/60 (0,1) e igual o superior a 3/60 (0,05).

Ceguera - agudeza visual inferior a 3/60 (0,05).

En este estudio la discapacidad visual moderada y la discapacidad visual grave están combinadas, agudeza visual inferior a 6/18 (0,3) e igual o superior a 3/60 (0,05). Comúnmente se reagrupan bajo el término “baja visión”.

Las personas que presentan algún tipo de limitación visual ya sea por pérdida parcial o total de la vista normalmente tienen ciertas dificultades para desplazarse, aún más si el entorno no es familiar para ellos ya que requieren algún tiempo de adaptación al nuevo entorno (Organización mundial de la salud, 2020).

ETA (Electronic Travel Aids).

Son las tecnologías asistidas cuyo propósito es mejorar la movilidad para las personas con algún tipo de discapacidad visual. Las ayudas electrónicas de viaje (ETA) ayudan a las personas con discapacidad visual a evitar obstáculos. Se trabajan e investigan las interfaces de usuarios y modos de interacción que permiten a los usuarios con discapacidad visual evitar obstáculos con la ayuda de muchos métodos como, por ejemplo, las pantallas virtuales acústicas, con las cuales se investiga las capacidades de diferentes sonidos y las alteraciones que producen en la parte psicológica del sujeto estas características de dicho sonido y con ello lograr obtener un sonido tridimensional.

ETA incluye simuladores con los cuales se logra avanzar en el estudio de diferentes experimentos e interfaces, con estos se facilita realizar la comparación entre diferentes aspectos y enfoques. Todas estas herramientas ayudan a poner a prueba una gran variedad de experiencias en mucho menos tiempo, con el objetivo de poder asistir a la gran demanda de herramientas de simulación y facilitar los procesos de investigación y desarrollo (Von Zabiensky y Bienhaus, 2017).

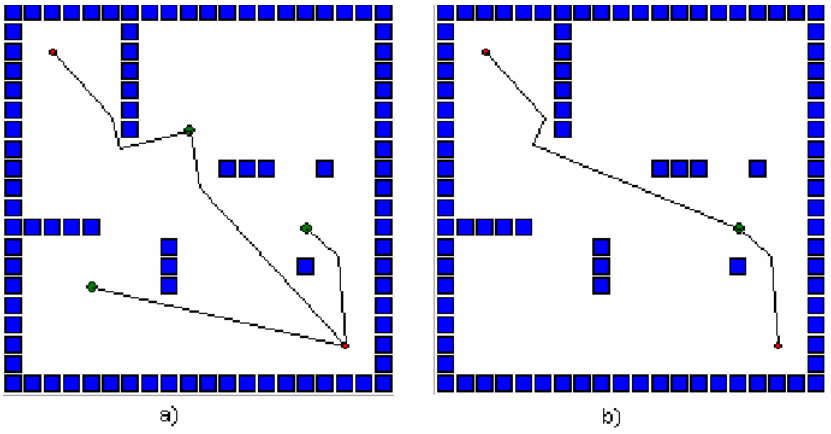
Algoritmo

Permite ejecutar un procedimiento y establecer un resultado, mediante instrucciones específicas a la solución de un problema. (Kelly, 2017). El proyecto "Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real" (Díaz et al., 2019), en su segunda etapa busca utilizar algoritmos para poder generar trayectorias optimas y mejorar el sistema inicialmente planteado.

Para evaluar el desempeño y la eficiencia de un algoritmo se necesitan tres parámetros fundamentales. En primer lugar, se trata de que el algoritmo utilice el menor tiempo posible al ejecutarse. En segundo lugar, hay que tener en cuenta el uso de memoria o recursos computacionales, se busca que sea el valor más bajo posible. En tercer lugar, se busca que, a pesar de los anteriores dos parámetros, su respuesta o resultado sea correcto o el algoritmo cumpla el propósito para el que fue diseñado en su totalidad. En la Figura 3 se observa a un robot que en un mismo espacio de trabajo recorre caminos diferentes, este es un indicador claro de la eficiencia del algoritmo utilizado.

Figura 3

Diseño y Simulación de un Robot



Fuente. tomado de Torres (2010, p.23).

Se pueden analizar muchos cuantificadores más, que permitirían determinar si el algoritmo posee un buen desempeño o no. Unos ejemplos serian el número de iteraciones, la cantidad de líneas de código o cantidad de instrucciones que se utilice para realizar ciertas acciones, pero todo esto se verá reflejado siempre en los tres principales parámetros.

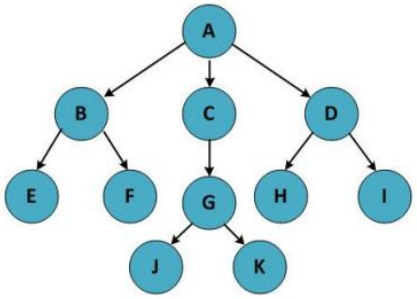
Grafos

Se pueden analizar muchos cuantificadores más, que permitirían determinar si el algoritmo posee un buen desempeño o no. Unos ejemplos serian el número de iteraciones, la cantidad de líneas de código o cantidad de instrucciones que se utilice para realizar ciertas acciones, pero todo esto se verá reflejado siempre en los tres principales parámetros.

Representan un conjunto de nodos unidos en una red. Si dos nodos están unidos, al recorrer de uno a otro se considerará sucesor el nodo al que nos movemos, y predecesor el nodo del que venimos. Además, habitualmente estará con un coste vinculado al desplazamiento entre nodos. La expansión de la búsqueda se realiza en forma de árbol. Partiendo del nodo inicial, se extenderá la búsqueda a sus nodos vecinos, de cada uno de estos nodos vecinos, a sus respectivos nodos vecinos, y así poder llegar al objetivo que se está buscando. La Figura 4 se presenta un grafo dirigido que está compuesto por 11 nodos y 10 aristas.

Figura 4

Grafo



Fuente. Elaboración propia del documento (2023).

Nodo

Es una parte fundamental para la composición de los grafos y es representado por un punto. Cada uno de estos vértices hace referencia a un evento o elemento que guarda cierta información del tema que el grafo esté evaluando (Kelly, 2017). Estos nodos pueden ser de diferentes tipos.

Nodo Padre

Se denomina padre al nodo del que parten una o más aristas con la función de unirlo a otros vértices diferentes. Se lo asocia al predecesor máximo del nodo.

Nodo Hijo

Estos nodos son aquellos que parten de un vértice anterior o un nodo padre. Estos pueden ser a su vez padres si poseen aristas que partan de ellos. También conocidos como cualquiera de los sucesores directos de un nodo.

Nodo Terminal

Este tipo de nodos no tiene más ramificaciones o aristas que partan de ellos, en otras palabras, estos nodos no serán padres y por esta razón se encuentran en los extremos de los grafos.

Nodos Vecinos

En el caso de las cuadriculas los nodos vecinos son aquellos que rodean al nodo padre, estos nodos pueden ser 8 o en caso de tomar otras diagonales pueden ser hasta 16 nodos.

Arista o Arcos

Son las líneas que unen a los nodos entre sí. Estas aristas suelen conocerse como *Edge* lo que hace referencia a los bordes y en algunos casos suelen tener dirección y también un valor

denominado ganancia de borde (*Gain Edge*) lo que ayuda a entregar información sobre este recorrido (Drozdek, 2007).

Camino

Es la secuencia de nodos que se encuentran conectados y forman una trayectoria desde un nodo del grafo hasta otro. El tamaño de este camino está dado por la cantidad de nodos que la compongan (Joyanes, 2006).

Características de los Algoritmos

Exactos

Los algoritmos deben ser capaces de lograr al menos una solución de un problema si ésta existe. En el caso del algoritmo *Wandering Motion Planner* (Kelly 2017) en su ejecución no garantiza la llegada al objetivo porque el éxito se considera cuestión de suerte: se observa que se escoge un camino tomando aleatoriamente posiciones sin importar que se hayan visitado o no, esto hace que puedan volver a ser tomados los mismos puntos. En particular este algoritmo necesita ser ejecutado varias veces para lograr conseguir una solución. En caso contrario, al avanzar en el estudio de algoritmos, encontramos a *Depth first*. Búsqueda en profundidad nos ayuda a encontrar elementos ligados en un grafo, e incluso encuentra caminos para cualquier par de vértices alcanzables entre ellos; sin embargo, estos caminos no son los más cortos (Rodríguez, 2018).

Sistemáticos

Presentan una secuencia clara y precisa para poder llegar a la solución. Un conjunto ordenado de procedimientos da como resultado una solución a un problema, un algoritmo debe seguir una secuencia de acuerdo con parámetros establecidos. Los algoritmos están orientados a crear alternativas de búsqueda para la solución de un problema para así ser comparados y poder

escoger el mejor algoritmo, no necesariamente encontrar de inmediato la solución óptima. Si comparamos los algoritmos A* cuya función es encontrar persistentemente y cuando se cumplan determinadas condiciones, el camino de menor costo entre un nodo origen y un punto destino, (Aragon, 2010) y el algoritmo de Dijkstra el cuál va explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo (Alcalde, 2017), encontramos que los pasos que sigue cada uno como algoritmos de búsqueda para encontrar el objetivo son sistemáticos y la diferencia se encuentra en que A* se mejora para hacer que la búsqueda sea más corta.

Finitos

Contienen un número determinado de iteraciones. Los algoritmos, según sea el objetivo o problema para solucionar deben dar una serie de pasos para ser ejecutados y debe evitar quedar en un bucle infinito o ciclo que no tenga fin (desbordamiento). A excepción de programaciones las cuales están dadas para que den resultados de un estudio dentro de un bucle infinito, pero ocasionalmente puede dar un resultado que muestre un desbordamiento que exija un cambio de código inmediato. En el caso de la programación en Arduino se tiene un bucle denominado *void loop* (Abellan, 2019) el cual es un ciclo infinito en él se ejecuta el programa y solo se termina cuando el dispositivo sea apagado, en cambio en el caso específico de un algoritmo de búsqueda donde el camino no encuentre la forma de llegar al punto final el programa caería en un desbordamiento debido a que el espacio de trabajo no encuentre pasos para seguir la búsqueda, esto requiere una corrección de código.

Concretos

Todo algoritmo debe dar solución de acuerdo con el problema planteado, teniendo en cuenta las funciones que debe cumplir. Lógicamente se trata de encontrar la solución esperada. Los algoritmos de búsqueda permiten obtener la ruta más corta entre un punto inicial y uno final por lo que resultan ser bastante útiles.

Admisibilidad

Si existen varios caminos que lleguen a la solución, un algoritmo retornará la solución óptima. Un algoritmo de búsqueda admisible es el que garantice el hallazgo de una ruta óptima entre el nodo de inicio y el nodo meta, si es que ella existe. En la búsqueda A* una heurística admisible es una que no sobreestima la distancia remanente entre el nodo presente y el nodo meta. Por ejemplo, siempre una ruta real entre dos ciudades es algo mayor y a lo sumo igual a la distancia en línea recta tomada de un mapa. Esta última distancia es así una heurística admisible pues en todo caso es "optimista", lo cual coincide con la definición (A* 2005).

Feacible o Factible

Se refiere a que logre llegar al objetivo de la mejor manera. En el caso específico de escoger una ruta pase por entornos libres, se desea que tratándose de algoritmos de búsqueda se garantice que al identificar el resultado el algoritmo sea capaz de reconocer obstáculos y bordes, lo que garantiza que la ruta que se va a escoger sea segura al permitir que no exista un truncamiento.

Otras Consideraciones

Tiempo

Cuánto demora el algoritmo en lograr obtener una solución. Hablaremos del tiempo de ejecución que significa el intervalo de tiempo en el que un programa se ejecuta. Los resultados

pueden cambiar sensiblemente de un computador a otro. Algunos ejemplos de los factores que pueden influir en el tiempo de ejecución son: Algoritmo usado, sistema operativo, velocidad del procesador, número de procesadores y conjunto de instrucciones que entiende, cantidad de memoria RAM, velocidad de cada una, coprocesador matemático, GPU. El tiempo de ejecución de un algoritmo va a ser una función que mide el número de operaciones elementales que realiza el algoritmo para un tamaño de entrada dado.

Costo Computacional

Cuanta memoria y capacidad del procesador necesita un algoritmo para ser ejecutado. Depende específicamente de tipo de computador que se utilice para generar un algoritmo. Tenemos ordenadores con propósitos generales que están diseñados para cubrir muchas necesidades, cómo el computador de escritorio que se utiliza si tiene datos almacenados, si los datos se encuentran dispuestos y para su prueba en el entorno no es necesario el desplazamiento. La capacidad computacional regularmente puede ser mejorada desde equipos quietos. También una Laptop si necesita moverse a un lugar donde se pueda ejecutar el sistema. En cambio, el Sistema embebido (Sanauja, 2011), son equipos que procesan datos digitalmente y son diseñados para realizar una tarea específica. Regularmente trabajan en tiempo real otorgando información de todo el sistema, su utilización se da cuando ya se tiene resultados, la implementación completa de un estudio, lo que requiere de una investigación muy exhaustiva, además de recursos económicos que permitan la adquisición de equipos computacionales completos.

Tiempo Real

Existen sistemas informáticos que tienen la capacidad de interactuar rápidamente con su entorno físico, ejecutan actividades o tareas en intervalos de tiempo bien definidos. Las tareas son ejecutadas inmediatamente en una forma sistemática.

Se desea que los algoritmos requieran del menor coste y otorguen el mayor beneficio así lograr la solución óptima del problema (Salcedo,2014).

Paralelizable

Es la propiedad que tiene un algoritmo de poder ser ejecutado por partes o secciones en el mismo instante de tiempo y por diferentes unidades de procesamiento para finalmente unir todas las secciones y poder obtener el resultado correcto. (Tinetti,2013)

Heurística

Este parámetro también es conocido como distancia entre dos puntos o en el caso de los grafos sería la distancia entre dos nodos. En el caso de grafos cuadriculados o matriciales podemos definir la heurística como la distancia en nodos que hay desde una posición a otra.

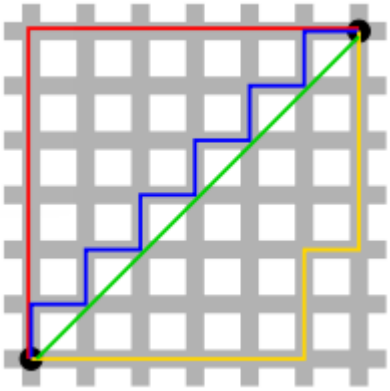
Esta distancia se la puede obtener de dos maneras y los métodos se conocen como heurística euclidiana y heurística de manhattan. La heurística euclidiana Está basada en el teorema de Pitágoras el cual expresa a la distancia entre los dos puntos como la hipotenusa, se calcula con la ecuación $H = \sqrt{a^2 + b^2}$ donde a y b son los catetos que forman el triángulo junto con la hipotenusa. Esta distancia en ciertos casos tiene la desventaja de no dar un resultado real, debido a que no se puede formar un camino o trayectoria totalmente diagonal entre dos puntos por que existirán muchos impedimentos u obstáculos que se encuentre por medio.

La heurística de Manhattan está diseñada para solucionar el inconveniente del método euclidiano y se basa en el ejemplo de la ciudad de Manhattan en donde no se puede llegar de un lugar a otro haciendo un recorrido diagonal por cuestión de las edificaciones, por este hecho se calcula la distancia más corta de un punto a otro realizando la suma de los catetos que hacen parte del triángulo formado por estos dos puntos y su intersección. Este valor es un dato mucho más adaptado a la realidad y es una gran ayuda al realizar análisis de recorridos o trayectos (eloviparo, 2018).

En la figura 5 se muestra en la línea roja la suma de los catetos, representando la heurística de Manhattan. Las líneas azul y amarilla, aunque no muestran la formación de un triángulo rectángulo, también establecen la misma distancia de la línea roja. Al contrario, la línea verde muestra la heurística euclidiana, donde se puede notar que tiene en cuenta las diagonales.

Figura 5

Distancia Manhattan



Fuente. tomado de Burrueco (2018, p.1).

Entornos o Ambientes

Para el caso de la planificación de rutas se conoce como entornos al espacio por el cual se calculará la trayectoria que es la unión de todos los puntos por los que pasa un objeto durante su movimiento. Se caracterizan dos tipos de entornos:

Entornos Estáticos.

En estos entornos no se presentan movimientos externos lo que quiere decir que los elementos o las características del entorno nunca cambian, por ejemplo, los obstáculos siempre permanecen en la misma ubicación. Es mucho más sencillo trabajar en estos ambientes debido a que se puede conocer toda la información del terreno y el calcular una trayectoria será un proceso que se deberá realizar solo una vez.

Entornos Dinámicos.

En estos casos se notará como el entorno podrá sufrir cambios en sus características por muchos factores, es más real porque siempre se sufren cambios por el movimiento de obstáculos u otros elementos que alteren la información que se tenía inicialmente. En la planificación de trayectorias se puede analizar estos casos de dos formas diferentes.

Uno es cuando las condiciones del espacio de trabajo fluctúan, lo más frecuente es notar como los obstáculos cambian de posición y esto hace que las rutas deban volverse a calcular ya que podrán arrojar un resultado diferente.

El segundo caso se lo comprende al analizar que el usuario no posee toda la información del ambiente y solo conocerá lo que está al alcance de sus sensores por lo que a medida que avanza se irá conociendo más información. También se tiene una fluctuación en el entorno y con ello se debe realizar una replanificación de la trayectoria para que se acomode a las nuevas condiciones.

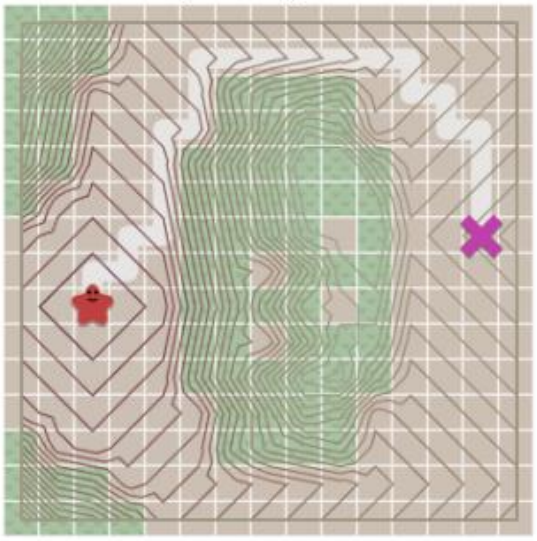
Para una mejor ejecución o análisis en estos entornos se debe tener en cuenta los dos casos de manera simultánea con lo cual la solución se podrá adaptar mucho mejor a los entornos reales a los que se está expuestos. El trabajar en este tipo de entornos será mucho más complicado por todas las variantes y condiciones que se deben analizar y con ello poder resolver cualquier tipo de situación que se pueda presentar (Yandún y Sotomayor,2017).

Terrenos

Una de las características que poseen los entornos es el tipo de terreno, esto hace referencia a como es la disposición del suelo o la superficie por donde se moverá el usuario. Cuando el terreno es plano, no existe ninguna diferencia en el coste o esfuerzo que hay para avanzar, todo posee un valor uniforme o constante que incluso se lo puede despreciar debido a que es una constante. Para terrenos no planos se encuentra muy presente el coste y es la dificultad que tiene cada espacio para dar un paso.

Figura 6

Comportamiento Algoritmo de Dijkstra Terrenos no Planos



Fuente. tomado de https://www.redblobgames.com/pathfinding/a-star

Se podrán encontrar diferentes tipos de terreno, por ejemplo, unas gradas y unas rampas, al hacer el análisis se debe tener en cuenta por cuál de los dos se podrá avanzar de manera más sencilla. Para estos casos la ruta óptima no solo consistirá en la más corta sino también en la que implique menos esfuerzo para el usuario. (Patel,2023). En la figura 6 de muestra el comportamiento del algoritmo Dijkstra en terrenos no planos.

Ciclo.

Son sentencias que ayudan a dar la orden de repetir instrucciones de manera cíclica hasta que se cumpla una condición la cual se encargará de hacer que dicho ciclo acabe y se pueda continuar ejecutando las siguientes líneas de código.

CPU y GPU

La unidad de procesamiento de gráficos, conocida como GPU, es en esencia un coprocesador que, en términos simples, opera de manera similar al CPU, pero su enfoque se centra en el procesamiento de gráficos. Su función principal es aliviar la carga de trabajo relacionada con la información gráfica, permitiendo así que la unidad central pueda realizar sus tareas de manera más eficiente. La distinción clave entre ambos reside en la arquitectura subyacente de estos dos componentes. Aunque su funcionamiento es similar en muchos aspectos, las GPU están diseñadas específicamente para llevar a cabo cálculos relacionados con gráficos de manera altamente eficiente. Esto las convierte en una opción óptima para su tarea principal, pero pueden no ser igual de competentes en otras áreas de procesamiento (Torres 2013).

El trabajar con estas unidades tiene una ventaja que consiste en la utilización con programación en paralelo o también conocida como paralelismo. Esta es una forma de computación que permite realizar cálculos o tareas de manera simultánea. Está basado en un principio fundamental de dividir los problemas que tienen un gran tamaño o una gran

complejidad en subproblemas más pequeños los cuales se pueden solucionar de una manera más rápida y también de forma simultánea con lo que se ejecutará de una manera más veloz y se obtendrán resultados muy eficientes.

Con la programación en paralelo de algoritmos secuenciales se logra utilizar el poder computacional al máximo de las tarjetas gráficas de una manera muy simple (Laguna et al., 2011).

Metodología

En este capítulo se muestra la metodología del proyecto investigativo, se plasmarán el tipo, diseño y enfoque de la investigación realizada, los objetivos con cada una de las actividades a realizar, mencionadas por pasos, finalmente se muestra el cronograma de actividades.

Tipo de Investigación

El proyecto de investigación contribuye al incremento de nuevos conocimientos y nuevas tecnologías, por esto se considera una investigación aplicada, apoyando las necesidades de nuestro entorno. Se plantea claramente los pasos que debe seguir cualquier investigador en cualquier ámbito para elegir un algoritmo de planificación de trayectorias. Centrando nuestro trabajo en el caso particular de personas invidentes se logró la implementación, mediante el software Matlab de siete algoritmos que proporcionan una guía para futuros trabajos.

En el proceso, se realizó una revisión de la literatura, tomando como referencia el caso particular de la problemática de las personas con discapacidad visual, con la finalidad de analizar las herramientas tecnológicas utilizadas para optimizar su movilidad, encontrando que en algunos asistentes se utiliza el módulo de planificación de trayectorias, dando lugar al estudio sistemático y su implementación de los algoritmos de planificación de trayectorias más utilizados.

Diseño Adoptado

La investigación tendrá un diseño exploratorio. La investigación exploratoria se lleva a cabo porque el tema que se requiere trabajar necesita ser comprendido profundamente, ya que en el mercado existen muy pocos asistentes a invidentes que presten el servicio de guía. Además, se han trabajado muy poco los algoritmos de planificación de trayectorias que faciliten la navegación por objetivo. También, el presentar una estructura absolutamente documentada le

dará herramientas a cualquier investigador para basar nuevos estudios en la serie de pasos planteados en este trabajo.

El objetivo de este método es explorar el problema y su contexto, lo que puede ayudar a las organizaciones o a los investigadores a ahorrar mucho tiempo y recursos, ya que permitirá determinar si merece la pena seguir adelante con un proyecto de investigación.

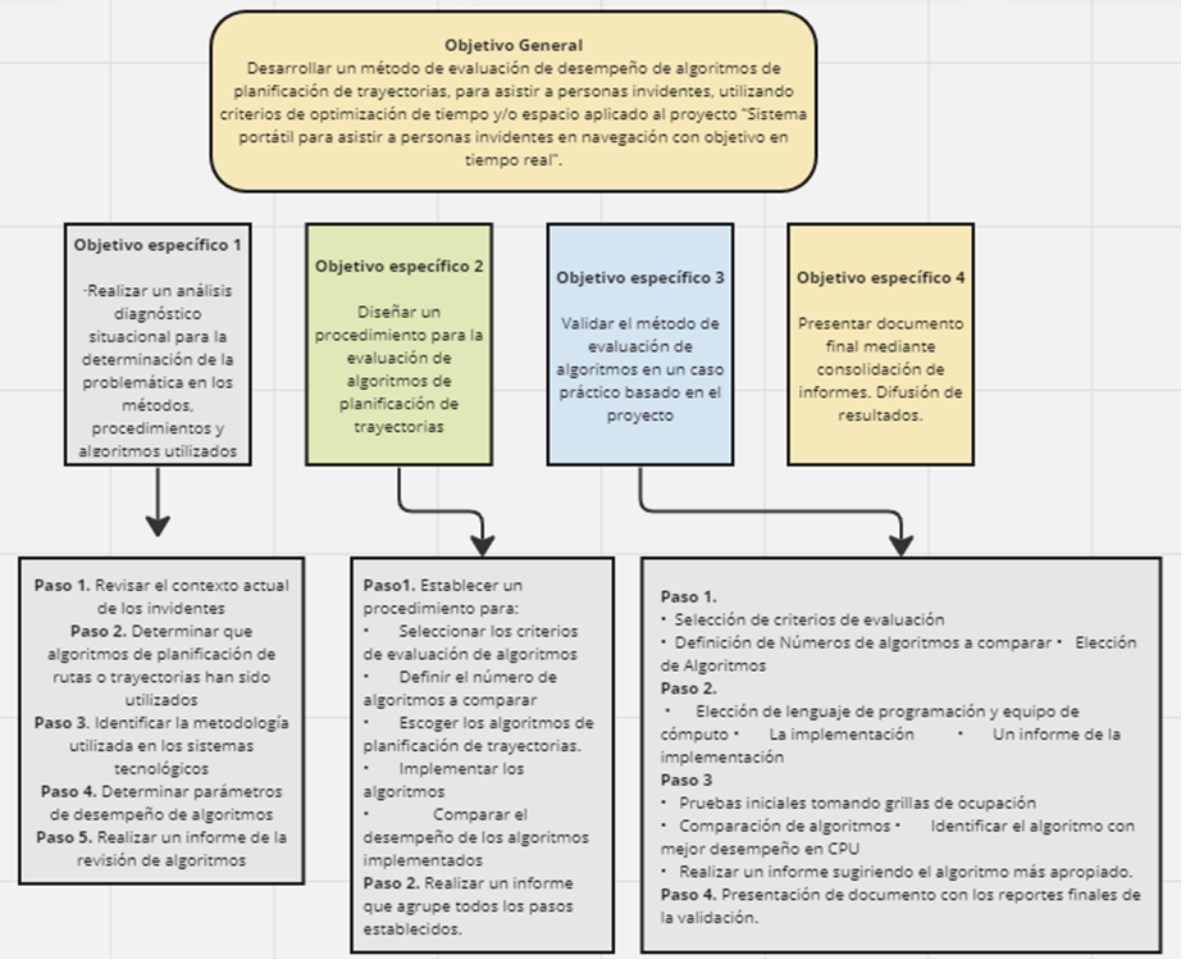
Enfoque

La propuesta investigativa descrita es de enfoque de tipo cuantitativa, (Hernández et al., 2003, p.5). Debido a que se usarán variables medibles y evaluables de manera numérica, que en el caso que corresponde estarán orientadas al rendimiento de los algoritmos de planificación de trayectorias previamente identificados y evaluados con el método propuesto. El estudio parte de un análisis de la efectividad de cada algoritmo. Al final de acuerdo con los parámetros que se definan en la investigación y en el método, se podrá seleccionar o sugerir el de mejor desempeño, según los criterios establecidos.

La metodología es la serie de pasos lógicamente estructurados y relacionados entre sí; que nos ayudarán a cumplir los 4 objetivos específicos planteados, de manera eficiente para alcanzar los resultados deseados. Se propone el diagrama de bloques de la figura 7. para sintetizar los objetivos expuestos en el anterior capítulo

Figura 7

Síntesis de los Objetivos Planteados y de los Pasos que se Van a Seguir para Lograr el Objetivo General de esta Investigación



Fuente. Elaboración propia del documento (2023).

En la Tabla 1 se muestra el cronograma de actividades, donde se propone un trabajo de investigación a 12 meses, el paso a paso apoya el proyecto para ir realizando cada actividad de acuerdo con cada uno de los objetivos planteados.

Tabla 1

Cronograma de Actividades

ACTIVIDAD	CRONOGRAMA DE ACTIVIDADES											
	MES 1	MES 2	MES 3	MES 4	MES 5	MES 6	MES 7	MES 8	MES 9	MES 10	MES 11	MES 12
Objetivo 1. Realizar un análisis diagnóstico situacional para la determinación de la problemática en los métodos, procedimientos y algoritmos utilizados actualmente para guiar a personas invidentes a través de trayectorias y las diversas metodologías que se usan en la evaluación mediante un análisis documental	X	X	X									
Paso 1. Revisar el contexto actual de los invidentes en Colombia y en el mundo	X											
Paso 2. Determinar que algoritmos de planificación de rutas o trayectorias han sido utilizados en dispositivos de asistentes a invidentes	X	X										
Paso 3. Identificar la metodología utilizada en los sistemas tecnológicos de ayuda a invidentes más sobresalientes	X	X										
Paso 4. Determinar parámetros de desempeño de algoritmos de planificación de trayectorias utilizados en la literatura.		X	X									
Paso 5. Realizar un informe de la revisión de algoritmos			X									
Objetivo 2. Diseñar un procedimiento para la evaluación de algoritmos de planificación de trayectorias para personas invidentes, utilizando criterios de optimización de tiempo y/o espacio.			X	X	X							
Paso1. Establecer un procedimiento para:												
• Seleccionar los criterios de evaluación de algoritmos de planificación de trayectorias			X	X	X							

• Definir el número de algoritmos a comparar • Escoger los algoritmos de planificación de trayectorias. • Implementar los algoritmos seleccionados. • Comparar el desempeño de los algoritmos implementados.						
Paso 2. Realizar un informe que agrupe todos los pasos establecidos	X					
Objetivo 3. Validar el método de evaluación de algoritmos en un caso práctico basado en el proyecto "Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real"		X	X	X	X	X
Paso 1. • Selección de criterios de evaluación de algoritmos de planificación de trayectorias • Definición de Números de algoritmos a comparar tomando cómo caso de estudio • Elección de Algoritmos de planificación de trayectorias.		X				
Paso 2. • Elección de lenguaje de programación y equipo de cómputo para implementar los algoritmos seleccionados. • La implementación de cada uno de los algoritmos • Un informe de la implementación de los algoritmos			X	X	X	
Paso 3 • Pruebas iniciales tomando grillas de ocupación en un punto de inicio y otro destino • Comparación de algoritmos implementados.						X

• Identificar el algoritmo con mejor desempeño en CPU • Realizar un informe sugiriendo el algoritmo más apropiado para asistir a personas invidentes			
Paso 4. Presentación de documento con los reportes finales de la validación.	X		
Objetivo 4. Presentación de documento con los reportes finales de la investigación. Difusión del método.		X	X

Fuente. Elaboración propia del documento (2023).

Contexto Actual de los Invidentes en Colombia y en el Mundo

En este capítulo se realizará un análisis diagnóstico situacional: el contexto actual de los invidentes en Colombia y en el mundo, algunos algoritmos utilizados actualmente para guiar a personas invidentes a través de trayectorias, diversas metodologías que se usan en la evaluación mediante un análisis documental y los parámetros de desempeño de algoritmos de planificación de trayectorias utilizados en la literatura.

En Colombia

La discapacidad visual se refiere a personas con deficiencias funcionales del órgano de la visión y, de las estructuras y funciones asociadas, incluidos los párpados. Está determinada por los niveles de deterioro de la función visual, y que se establece tras la medición de la agudeza visual y del campo visual de cada uno de los ojos por separado. (Martínez, 2000).

El Censo 2018 arrojó un resultado de acuerdo con la escala de medición de la discapacidad del WG, en el CNPV 2018, de 1´948.332 personas con discapacidad visual equivalente al 62.17% de la población con discapacidad en Colombia, de un total de 3´134.036 personas con discapacidad, en general equivalente al 7.1% de la población colombiana. Estudios muestran que, en Colombia, hay aproximadamente 296.000 personas totalmente ciegas. (Dussan, 2020). En la Figura 8 se observa a personas con discapacidad visual capacitándose en cuestiones educativas, la imagen es tomada del catálogo de servicios del instituto Nacional para ciegos INCI. Este instituto ofrece diferentes servicios para garantizar el aprendizaje continuo, el acceso a la información, el entretenimiento y la accesibilidad de la población con discapacidad visual, docentes, agentes educativos, cuidadores y ciudadanía en general.

Figura 8

Población con Discapacidad Visual

Fuente. Catálogo de Servicios · Instituto Nacional Para Ciegos –INCI (2021).

Políticas Públicas

INCI es el Instituto Nacional para Ciegos, una entidad de carácter técnico asesor adscrita al Ministerio de Educación, creada mediante el Decreto 1955 del 15 de Julio de 1955. Desde su creación el INCI trabaja para garantizar los derechos de los colombianos ciegos y con baja visión en términos de inclusión social, educativa, económica, política y cultural. Brinda servicios de asistencia técnica y asesoría a las demás entidades que a nivel nacional, territorial y local tienen a cargo la atención de las personas con discapacidad visual en el país.

Educación

En Colombia, El Instituto Nacional para Ciegos –INCI brinda asesoría y acompañamiento de manera presencial y virtual a las entidades territoriales, en coordinación con las secretarías de educación departamentales y municipales. Realiza capacitación a los docentes de las instituciones educativas que atienden estudiantes con discapacidad visual en áreas tiflológicas, estrategias pedagógicas, baja visión y tecnología especializada.

El Instituto Nacional para Ciegos –INCI, está trabajando en el desarrollo jurídico del Braille en Colombia, busca reglamentar su uso y estandarizar su escritura. Lo realiza mediante 5 frentes de trabajo paralelo y articulado. Se creará un curso de Braille, por parte del SENA y certificará en estas competencias.

Publicaciones de Ayudas a Invidentes

Se han realizado programas radiales en la emisora virtual INCIRadio, acerca de las temáticas de discapacidad. En 2021 se hicieron 3 contenidos audiovisuales con descripción en audio y narrativas accesibles para las personas con discapacidad visual. El fortalecimiento en el uso de herramientas virtuales disponibles en la entidad ayudará para continuar afrontando los desafíos laborales que se presentan por la restricción causada por la emergencia sanitaria y ambiental.

En el Mundo

En el mundo hay al menos 2200 millones de personas con deterioro de la visión cercana o distante. En al menos 1000 millones de esos casos, es decir, casi la mitad, la discapacidad visual podría haberse evitado o todavía no se ha aplicado un tratamiento (Organización Mundial de la Salud, 2022).

El Código Internacional de Enfermedades CIE-10 define la ceguera “como una agudeza visual menor de 0.05 (20/400, 3/60, 1.3 logMAR), o una correspondiente pérdida del campo visual menor de 10 grados en el mejor ojo con la mejor corrección posible”. Por discapacidad visual grave se entiende una agudeza visual inferior a 20/200, 6/60 e igual o superior a 3/60 o 20/400, y por discapacidad visual moderada, una agudeza visual de entre menos de 6/18 (20/60) y 6/60 (20/200) (OMS, *Definitions of blindness and visual impairment).*

A nivel mundial, las principales causas de discapacidad visual son:

errores de refracción no corregidos

catarata

la degeneración macular relacionada con la edad

glaucoma

retinopatía diabética

opacidad corneal

tracoma

De acuerdo con la OMS, el reconocimiento de la creciente necesidad proporcionará atención oftalmológica en todo el mundo. La Figura 9 es tomada de la página oficial de la OMS, donde dentro de las actividades está la guía de Integración de la atención oftalmológica en los sistemas de salud, “Atención oftalmológica integrada centrada en las personas (IPEC), incluidas las deficiencias visuales y la ceguera prevenibles”.

Figura 9

Integración de la Atención Oftalmológica en los Sistemas de Salud

Fuente. Tomado de OMS / NOOR / Sebastián Liste. https://www.who.int/activities/integrating-eye-care-in-health-systems---guide-for-action.

Impacto Personal al Tener Discapacidad Visual

Los niños pequeños con discapacidad visual severa de aparición temprana pueden experimentar retraso en el desarrollo motor, del lenguaje, emocional, social y cognitivo, con consecuencias de por vida. Los niños en edad escolar con discapacidad visual también pueden experimentar niveles más bajos de logros educativos.

La discapacidad visual afecta gravemente la calidad de vida de las poblaciones adultas. Los adultos con discapacidad visual a menudo tienen tasas más bajas de participación y productividad en la fuerza laboral y tasas más altas de depresión y ansiedad. En el caso de los adultos mayores, la discapacidad visual puede contribuir al aislamiento social, dificultad para caminar, un mayor riesgo de caídas y fracturas y una mayor probabilidad de ingreso temprano a hogares de ancianos o de cuidados.

Impacto Económico al Tener Discapacidad Visual

La discapacidad visual representa una enorme carga financiera mundial. Por ejemplo, se estimó que los costos globales anuales de las pérdidas de productividad asociadas con la discapacidad visual por miopía y presbicia no corregidas ascendían a 244 mil millones de dólares y 25,4 mil millones de dólares, respectivamente, de acuerdo con cifras de la Organización Mundial de la Salud (OMS).

Herramientas Tecnológicas

Las herramientas tecnológicas pueden ayudar a facilitar la vida diaria de las personas con problemas de visión. Ya sea que necesite más ayuda para navegar a un destino, leer o disfrutar de otra de sus actividades favoritas, la tecnología puede ayudarlo. (Mukamal, 2021). Algunos ejemplos de las mejores aplicaciones, dispositivos y recursos para personas con baja visión se

pueden usar en un teléfono inteligente o tableta, incluyendo la mayoría de los dispositivos Android e iOS.

Herramientas Integradas en Dispositivos Apple

VoiceOver es un lector de pantalla que proporciona una descripción de todo lo que sucede en su pantalla. Siri es un sistema de reconocimiento de voz que le permite enviar mensajes, realizar llamadas telefónicas y mucho más.

A Herramientas Integradas en Dispositivos Android

Las aplicaciones que se ejecutan en el sistema operativo Android utilizan un lector de pantalla integrado llamado TalkBack.

Los dispositivos Android también usan *Google Assistant*, un "ayudante virtual" impulsado por inteligencia artificial. El *Google Asistant* le permite realizar una amplia variedad de acciones en su teléfono o tableta usando comandos de voz.

Para Tareas Diarias e Identificación de Objetos

Seeing AI, esta aplicación puede narrar el mundo que lo rodea. Simplemente apunte la cámara de su teléfono o iPad a algo, y *Seeing* AI le dirá en voz alta qué es. Puede ayudar a leer la moneda, el nombre de los colores e incluso a descifrar la letra cursiva manuscrita.

Lookout, proporciona comentarios hablados sobre las cosas que lo rodean. *Lookout* usa la cámara y los sensores de su dispositivo para reconocer texto, personas y objetos.

Be My Eyes aplicación que conecta a personas con discapacidad visual con voluntarios videntes a través de una videollamada en vivo. Los voluntarios “prestan sus ojos” para ayudar con tareas breves y sencillas, como leer un letrero en la calle o solucionar problemas de tecnología. Por razones de seguridad, no pida a los voluntarios que lean información personal como el correo que contenga su dirección, información financiera o de su tarjeta de crédito.

Para Navegación y Transporte

Soundscape. diseñada para usar con auriculares estéreo, esta aplicación proporciona señales de audio en 3D de los alrededores en tiempo real. Se puede utilizar junto con la navegación GPS.

Nearby explorer, esta aplicación combina navegación GPS, indicaciones paso a paso en modo peatonal y de vehículo, puntos de interés y datos de tránsito. *RightHear*, una aplicación de orientación espacial creada para personas con baja visión proporciona información de navegación y puntos de interés mediante guías de audio.

Para Socializar

Facing Emotions, aplicación de reconocimiento facial que utiliza inteligencia artificial (IA) para identificar siete emociones diferentes convirtiéndolas en sonidos que puedes escuchar. La aplicación debe usarse con el Mate 20 Pro de Huawei.

Instagram, popular aplicación de redes sociales ahora incluye una función de texto alternativa que permite a los usuarios agregar y escuchar descripciones detalladas de fotos mientras navegan. Esta función trabaja junto con el lector de pantalla del dispositivo.

YouTube para Android, ha ampliado sus funciones de comando de voz para la navegación por video.

Pandora, recientemente se agregó el comando de voz "Hola Pandora", para diversas tareas.

Algoritmos de Planificación de Rutas o Trayectorias que Han Sido Utilizados en Dispositivos de Asistentes a Invidentes.

An Indoor Navigation System for Visually Impaired People Using a Path Finding Algorithm and a Wearable Cap.

Consiste en una investigación donde existe una implementación de un dispositivo cuya función principal es otorgarle a la persona invidente información sobre el espacio por el cual está transitando y además guiarlo por una ruta calculada hacia su destino.

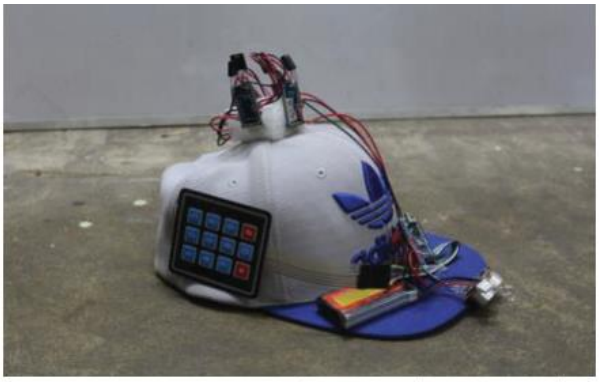
Este estudio pudo demostrar la hipótesis, que a la persona le ayuda mucho más una información completa entregada por el dispositivo que la información que proviene por medio de la voz. Esto se puede observar en resultados como el tiempo en el que se completó el trayecto calculado desde el inicio hasta el final y también la distancia recorrida. Todas las conclusiones se obtuvieron cuando se puso a prueba los dos casos, con el dispositivo y la orientación por voz, con personas vendadas de los ojos y también personas realizando de espaldas el recorrido.

En el documento no se especifica cual es algoritmo utilizado para realizar el cálculo de las trayectorias, pero se puede afirmar que es un algoritmo para entonos estáticos, esto se deduce por las pruebas que se realizaron, siempre en entornos interiores o conocidos.

El objetivo principal de esta investigación es proporcionarles a las personas que sufren esta discapacidad visual un dispositivo portátil y compacto, mostrado en la figura 10, que les permita conocer la posición actual en donde se encuentran posicionados, entregarles una información clara sobre el entorno en el que se encuentran y direccionarla hacia el destino con lo cual el usuario podrá ser capaz de tener acceso, obtener una mayor independencia y con esto tener una mejor calidad de vida. (Islam, 2018).

Figura 10

Prototipo Diseñado de Gorra



Designed prototype of wearable cap for th

Fuente. Tomada de 3rd International Conference for Convergence in Technology (I2CT), (2018, pp. 1-6).

Wearable RGBD Indoor Navigation System for the Blind.

Trabaja el desarrollo de un sistema que permite ayudar a una persona invidente a transitar con un poco más de libertad por medio de su entorno y así poder hacer que sea más independiente.

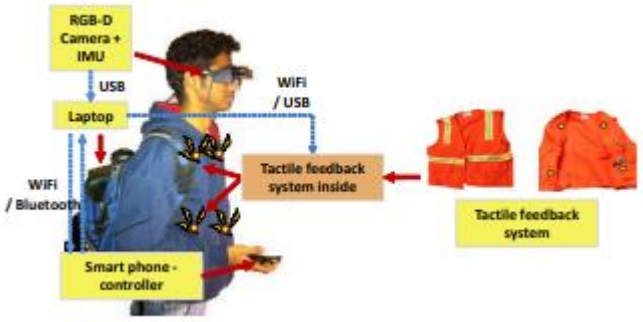
Este novedoso sistema de navegación mostrado en la Figura 11, posee una cámara RGBD que capta imágenes en color y además mide la profundidad asociada a los píxeles de la imagen. Crea un mapa en tres dimensiones del entorno con lo cual procede a hacer un análisis de las opciones que tiene para transitar.

Después de analizar toda la información de entrada se crea una ruta en el cual se encuentra el usuario hasta donde se quiere llegar con la ayuda del algoritmo D star lite, el cual está perfectamente diseñado para procesos en tiempo real como los que plantea la investigación. Para guiar al usuario hace uso de una retroalimentación háptica, esto consiste en dar información por medio del tacto. Utiliza motores vibradores los cuales están ubicados de manera que su

vibración le indique a la persona con discapacidad visual hacia donde debe dirigirse o el sentido en el que debe caminar para seguir el trayecto calculado y poder así evitar los obstáculos en tiempo real. También utiliza la retroalimentación de forma auditiva para proporcionar otra manera de guiar al usuario, en este caso se tienen comandos con palabras clave como “derecha” o” izquierda” con las cuales él podrá tomar la decisión de la acción que debe realizar en dicho momento del recorrido. Toda esta información, tanto háptico como sonora, es cuidadosamente diseñada para que la carga cognitiva sea razonable y así no afectar a la persona. (Lee, 2015).

Figura 11

Sistema de Navegación Interior Portátil RGBD para Ciegos



Fuente. Tomado de Lee (2015, p.2).

CCNY Smart Cane

Este dispositivo consta de dos componentes principales: un dispositivo móvil (teléfono o tableta) habilitado para Google Tango que se monta en el pecho del usuario y un bastón blanco robótico en su mano.

El dispositivo móvil se utiliza para reconocer una ubicación en interiores y proporcionar una guía de navegación a través de audio al usuario a un destino específico para lo cual se calcula una trayectoria por medio del algoritmo A star, el cual garantiza una ruta dentro del espacio conocido en entornos interiores y estáticos.

El dispositivo móvil también se comunicará con el bastón blanco robótico a través de Bluetooth para guiar a un usuario con discapacidad visual en consecuencia. Estas indicaciones se las consigue por medio de estímulos o realimentación háptica con lo cual la persona con esta incapacidad podrá tomar la decisión de los movimientos que debe hacer para llegar a su objetivo de manera segura (Chen, 2017).

ISANA: Wearable Context-Aware Indoor Assistive Navigation with Obstacle Avoidance for the Blind.

Consiste en un artefacto diseñado para ayudar a un usuario que sufra una discapacidad visual, el cual se ejecuta en un dispositivo móvil tableta Android Tango portátil, que tiene funciones integradas de archivo de descripción de área. La configuración física muestra un dispositivo Tango, un soporte y un bastón blanco. ISANA proporciona a los usuarios ciegos un conocimiento del contexto de alto nivel basado en mapas semánticos de interiores y navegación con funcionalidades para evitar obstáculos, junto con una interacción de voz y audio fácil de usar. Los mapas de interiores proporcionan modelos espaciales del entorno, así como perfiles de usuarios. La cámara RGBD admite el reconocimiento de escenas y la detección de obstáculos. La navegación asistida genera la ruta con la ayuda del algoritmo A star e interactúa con el usuario mediante una interfaz de voz y audio. El usuario invidente es el que toma la decisión final, ya que ISANA responderá al movimiento del usuario, en lugar de controlarlo. Además, el usuario ciego todavía puede, como de costumbre, usar un bastón blanco para obtener información táctil para la zona de seguridad en el frente. Hay dos aplicaciones del sistema, el editor de mapas de interiores de ISANA que realiza la construcción de mapas semánticos fuera de línea mediante el análisis de archivos de modelos arquitectónicos, y realiza las funcionalidades de localización

semántica en el mapa semántico, guía de navegación, evitación y alerta de obstáculos y HMI de audio para el usuario (Li, 2016).

Human-Robot Interaction for Assisted Wayfinding of a Robotic Navigation Aid for the Blind.

Consiste en una ayuda para las personas invidentes que tiene dos funciones fundamentales. En primer lugar, se encuentra la orientación asistida, donde es necesario calcular una trayectoria y esta tarea se la realiza haciendo uso del algoritmo A star y también desarrollo un nuevo tipo de estimación de la posición del elemento mientras navega (odometría) de tipo inercial y visual la cual estima la posición. Para lo anterior se utilizaron recursos como una cámara RGBD, que proporciona datos de imagen y profundidad, y también una unidad de medición inercia o IMU (*inertial measurement unit*), con la que se proporcionan datos como la velocidad, orientación, fuerzas gravitacionales, aceleración, entre otras.

Con todos los datos y el resultado de la trayectoria calculada se tiene una interfaz de interacción humano-robot con dos modos de guía. Uno de los modos utiliza una punta rodante motorizada la cual orienta la dirección del viaje deseada a la cual el usuario seguirá la ruta planificada, el segundo modo de guía se asemeja a un bastón blanco donde el dispositivo utilizará la interfaz de audio para dar las señales a dicha persona mediante mensajes de voz. El método tiene la capacidad de detectar la intención del usuario y automáticamente seleccionar el más apropiado según las intenciones detectadas por el sistema. (Zhang, 2019).

A Blind Aid System Based on Jetson TX2 Embedded System and Deep Learning Technique.

Es una propuesta de investigación e implementación de un dispositivo que ayudará a las personas invidentes a transitar o caminar con más seguridad. Este artefacto está ubicado en la cabeza del usuario y su función principal es reconocer los obstáculos que se presenten en el camino de la persona con la ayuda de la inteligencia artificial. A diferencia de otros proyectos de

estudio o dispositivos no planifica una trayectoria o calcula un camino a un objetivo determinado por lo cual no hace uso de ningún algoritmo de planificación. Su función es informar por medio de audio que existe un obstáculo, qué tipo de obstáculo es y la posición donde se encuentra este objeto con el fin de que la persona pueda saber y evadir dicho peligro y continuar su camino (Mukamal, 2021).

Las investigaciones tienen en común unos parámetros para evaluar su desempeño. En primer lugar, deben garantizar que se calcule un camino seguro para el usuario. En segundo lugar, se busca que los sistemas puedan operar en tiempo real, lo que garantizaría al usuario la seguridad de poderse desplazar por los entornos, siendo apoyados por cada uno de los sistemas mencionados. Por último, se necesita que los instrumentos posean un alto rendimiento y puedan funcionar con el más bajo costo computacional, esto garantiza una respuesta rápida en cuestión de tiempo y también ayuda a que el producto o prototipo pueda ser lo más económico posible. En la Tabla 2 se muestra el resumen de las características principales de los dispositivos estudiados que tienen que ver con asistentes a invidentes. Y en la Tabla 3 se observa el algoritmo utilizado en cada uno de estos dispositivos.

Tabla 2

Síntesis de los Dispositivos de Asistentes a Invidentes Contemplados en la Literatura

Artículos	Sensor	Realimentación	Desempeño	Procesador	Funcionalidades de los algoritmos	Contribución	Limitaciones
An Indoor Navigation System for Visually Impaired People Using a Path Finding Algorithm and a Wearable Cap	Transmisores de infrarrojos, receptores de infrarrojos y auriculares. sensor ultrasónico HR-04	Auditiva por medio de auriculares	Tiempo real	Arduino Nano, Arduino Pro Mini	Algoritmo de ruta más corta realizando una detección de obstáculos a medida que aparecen en el camino.	sistema de navegación interior mediante el uso de un algoritmo de búsqueda de ruta y una gorra portátil instalada con receptores IR y sensor ultrasónico	El número de transmisores de infrarrojos dependerá del tamaño de la habitación. No es útil en el caso de ser un ambiente abierto por la inexistencia de los trasmisores lo cual deja con poca

							información para realizar la búsqueda del camino.
Wearable RGBD Indoor Navigation System for the Blind.	Sensor IMU MEMS de 9 DOF en la parte superior del sensor RGB-D	Háptica por medio de motores de micro vibración	Tiempo real	Interfaz de usuario de teléfono inteligente, un dispositivo de cámara RGBD	Algoritmo de estimación de movimiento, mapeo, detección de obstáculos y planificación de rutas en tiempo real. D* lite	Navegación segura en entornos interiores. Mejora el rendimiento de movilidad	Las personas visitan algunos lugares varias veces.
CCNY Smart Cane	Google TANGO, IMU (unidad de medición inercial),	Auditiva a través de parlantes y Háptico por medio de motores de vibración,	Tiempo real	Dispositivo móvil (teléfono o tableta) habilitado para Google Tango	Software de navegación para planificar una ruta desde la ubicación del usuario hasta cualquier punto de ruta dentro de un entorno interior. A*	El software en el dispositivo móvil es capaz de comunicarse con el bastón blanco robótico para planificar una ruta y	Precisión de la posición y rotación del bastón

					llevar a un usuario de BVI a un destino dentro de un ambiente interior.	
I SANA: Wearable Context-Aware Indoor Assistive Navigation with Obstacle Avoidance for the Blind	t ableta Android Tango portátil, Cámara de profundidad infrarroja (IR) de 5HZ. Cámara RGB-Depth	HMI de audio y de voz	Ti empo real	t ableta Android móvil integrada con cámara RGB-Depth	Alg oritmo planificador de ruta para encontrar la ruta a un destino deseado A*	Pr oporcionar a la comunidad ciega una herramienta de ayuda para viajes en interiores, conciencia del contexto y conciencia de la situación del mundo ciber físico circundante del usuario.	P redicción de obstáculos dinámicos y en la comprensión del entorno.
H uman-Robot Interaction for	S ensores IMU.	Audio por medio de parlante audífono y háptica por	Ti empo real	I magen y los datos de profundid	A* Algoritmo de planificación de rutas	La búsqueda de caminos y la interacción humano-	

Assisted Wayfinding of a Robotic Navigation Aid for the Blind	cámara RGB-D	medio del bastón.		ad de una cámara RGB-D		robot (HRI), para la búsqueda de caminos asistida
A Blind Aid System Based on Jetson TX2 Embedded System and Deep Learning Technique	C ámara	Audio por medio de parlante audífono.	Ti empo real	S istema portátil de ayuda para ciegos, que utiliza el sistema integrado Jetson TX2 y la técnica de aprendizaje profundo.	Bas ado en la inteligencia artificial el cuál identifica, clasifica y ubica objetos a través de redes neuronales convolucionales profundas.	La s personas ciegas sepan con certeza qué tipo de objetos se encuentran frente a ellos, lo que puede ayudarlos a caminar de manera segura y conveniente por sí mismos.	Ú nicamente identifica y alerta al usuario de los objetos que están a su alrededor, pero no planea una ruta para ayudar al invidente a llegar de un lugar a otro.

Fuente. Elaboración propia del documento (2023).

Tabla 3

Algoritmos de Planificación de Trayectorias de Asistentes a Invidentes Contemplados en la Literatura

Dispositivo	Algoritmo
An Indoor Navigation System for Visually Impaired People Using a Path Finding Algorithm and a Wearable Cap	Algoritmo no especificado
Wearable RGBD Indoor Navigation System for the Blind.	D* lite
CCNY Smart Cane	A*
ISANA: Wearable Context-Aware Indoor Assistive Navigation with Obstacle Avoidance for the Blind	A*
Human-Robot Interaction for Assisted Wayfinding of a Robotic Navigation Aid for the Blind	A*
A Blind Aid System Based on Jetson TX2 Embedded System and Deep Learning Technique	No posee algoritmo de planificación de trayectorias

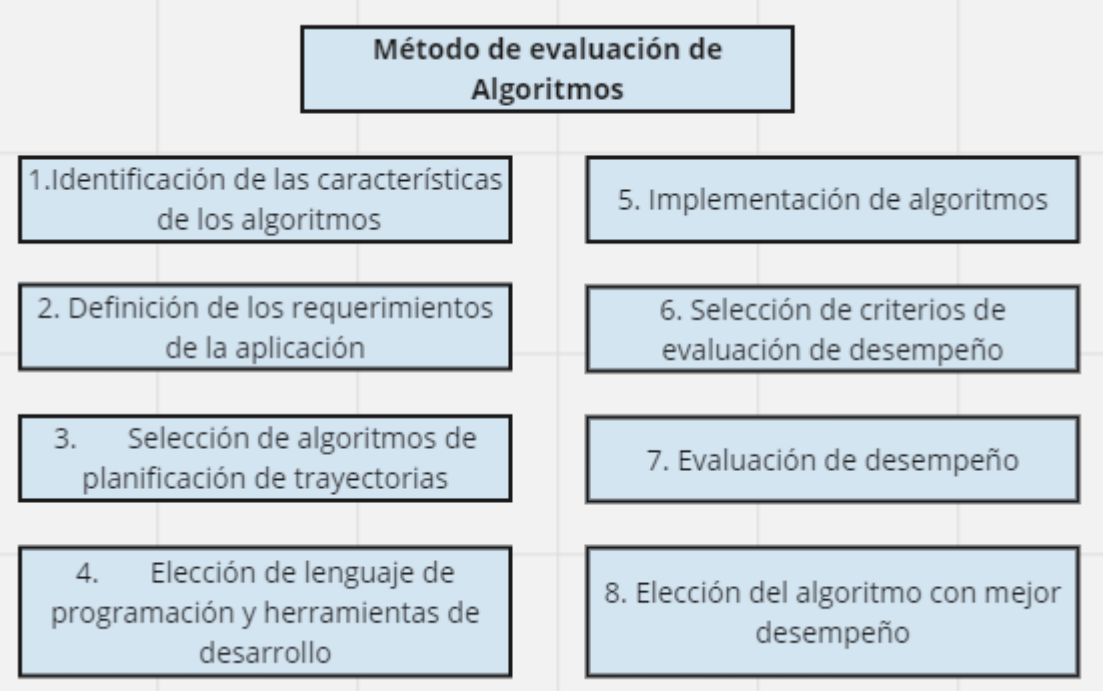
Fuente. Elaboración propia del documento (2023).

Método Propuesto

Este capítulo hace referencia al desarrollo de los objetivos 2 y 3, donde se diseña y se valida el método de evaluación de algoritmos en un caso práctico basado en el proyecto “Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real”. En la Figura 12 se presenta la secuencia de pasos que permitirá realizar una evaluación sistemática de algoritmos de planificación de trayectorias. Se dan detalles de cada actividad: características de los algoritmos, requerimientos de la aplicación, selección de los algoritmos y del lenguaje de programación, implementación de algoritmos, selección de criterios y evaluación del desempeño, por último, la elección del algoritmo con mejor desempeño.

Figura 12

Método Propuesto



Fuente. Elaboración propia del documento (2023).

Características de los Algoritmos

Lo que se desea es aplicar algoritmos de planificación de trayectorias y determinar el camino óptimo, en entornos dinámicos y que sea utilizado en tiempo real, implementando el sistema en un software capaz de ser el soporte lógico para realizar una secuencia de movimientos, y de esta manera, una persona pueda movilizarse de manera segura en un entorno desconocido. Existen muchos modelos y algoritmos que se han creado para generar rutas a través de códigos que analizan y ejecutan movimiento. La intención de este proyecto es analizar el desempeño de algunos de ellos. A continuación, se mencionan algunos algoritmos que se estudiaron en el desarrollo del proyecto.

Wander Planner

No es exacto debido a que no es seguro que se encuentre el objetivo, es cuestión de suerte el encontrar la meta.

El algoritmo es sistemático ya que tiene un paso a paso definido para realizar la tarea.

Wander Planner no asegura encontrar la solución, su manera de buscar depende del azar y también no tiene un proceso bien planteado para encontrar el punto final.

Mientras se observa el comportamiento del algoritmo a medida que aumenta el tamaño del espacio de trabajo es más difícil que el algoritmo finalice ya que no encuentra el destino, por esta razón no se lo puede considerar un algoritmo finito.

Este algoritmo no es capaz de escoger la mejor solución, el algoritmo se queda con la primera trayectoria que pueda encontrar. Debido a lo anterior se lo considera no admisible.

El algoritmo es factible porque es capaz de reconocer obstáculos para no pasar por ellos a medida que calculan o realizan el recorrido.

El tiempo de ejecución de este algoritmo es comparativamente alto porque el camino puede tener rutas cíclicas.

No posee un costo computacional muy alto ya que no debe tener en cuenta cuales posiciones han sido revisadas y cuáles no.

No se adecua a un entorno dinámico.

No se puede realizar el proceso de manera paralela.

Spanning Tree

Es exacto, obtiene una trayectoria entre dos puntos sin tener recorridos cíclicos pero la solución no siempre será la óptima o más corta ya que su proceso sigue dependiendo del azar y el resultado son trayectorias impredecibles.

Posee unos pasos definidos para dar un resultado, por eso es un algoritmo sistemático.

Este algoritmo tiene en cuenta las casillas que ya fueron visitadas. Hay casos en los que el algoritmo se desborda porque no encuentra posiciones disponibles para ir.

No es capaz de escoger el mejor camino, sólo se queda con la primera solución que encuentra.

Spanning tree también puede diferenciar entre casillas disponibles, ya visitadas y obstáculos.

Aunque este algoritmo ya no realiza rutas cíclicas el tiempo de ejecución es relativamente alto por la forma como escoge las direcciones o las casillas.

Su costo computacional es un poco mayor al anterior algoritmo porque es necesario recordar o guardar las casillas que ya han sido visitadas y las que se encuentran disponibles para dar cada paso.

En su proceso siempre considera que el entorno es estático, no modifica su trayectoria para un entorno dinámico.

No se puede realizar el proceso de manera paralela.

Breadth First

Este algoritmo es exacto porque siempre encuentra el camino hasta la meta.

Se realizan unos pasos bien definidos para realizar la búsqueda del camino los cuales aseguran encontrar la solución del problema.

El proceso que realiza este algoritmo es basado en una teoría que ayuda a garantizar que sea posible encontrar un camino del punto de inicio al punto final.

Se puede considerar que es un algoritmo finito, realiza un número determinado de iteraciones y no tiene probabilidad de desbordamientos o de realizar caminos cíclicos.

Es capaz de optar por la mejor solución de las muchas que existen para encontrar la trayectoria.

El algoritmo es factible porque puede detectar en qué posiciones existen obstáculos y logra evitarlos.

El tiempo de ejecución de este algoritmo es mucho mayor a los anteriores debido al proceso que debe realizarse para encontrar la mejor solución.

También hay un aumento en el costo computacional porque necesita más cantidad de memoria para almacenar más datos y se le exige más al procesador para realizar las tareas.

Debido al proceso que utiliza para calcular el camino no es posible que se realice en un entorno dinámico.

No se puede realizar el proceso de manera paralela.

Dijkstra

Este algoritmo también garantiza que se encuentra una trayectoria entre los dos puntos que se requieren.

Los pasos son definidos para poder encontrar esta solución de la mejor manera posible.

Maneja la misma teoría que el anterior algoritmo para poder llegar al objetivo de una manera más clara y concisa.

Al evitar desbordamientos en el código y que no se realicen trayectorias en círculos innecesarias se puede garantizar que el algoritmo sea finito ya que después de dar una cantidad de pasos siempre llega al final.

El algoritmo de *Dijkstra* también debe tener en cuenta el coste que tiene cada casilla y por ello el proceso ayuda a que se escoja el mejor camino teniendo en cuenta todas esas características.

Este algoritmo no sólo diferencia los obstáculos sino también analiza el coste que tiene cada posición lo cual hace que esté más completo el proceso para encontrar la mejor ruta.

En cuestión al tiempo de ejecución del programa es un poco mayor al anterior algoritmo ya que ahora debe analizar el dato de los costes en cada posición.

El coste computacional también es mayor ya que se debe guardar además los datos de cada coste.

Al igual que el algoritmo anterior, funciona solo para entornos estáticos.

No se puede realizar el proceso de manera paralela.

A*

Este algoritmo siempre encuentra la solución al problema.

Tiene pasos bien planteados para encontrar el camino entre los dos puntos.

El proceso para encontrar la trayectoria es planteado y fundamentado con la teoría que maneja el algoritmo de *Dijkstra* y otros argumentos para poder encontrar esta solución.

Se puede catalogar como finito porque asegura que no se presentan desbordamientos y debido a que siempre encuentra un camino siempre tiene un fin al realizar las iteraciones.

A pesar de que existen muchos resultados para la trayectoria entre dos puntos este algoritmo escoge siempre la óptima.

También se tiene en cuenta los obstáculos y costes en este algoritmo para llegar a una respuesta óptima.

El tiempo de ejecución es menor que en los algoritmos *Dijkstra* y *breadth first* ya que recorta el proceso que realizan estos últimos para encontrar el camino.

Se intenta reducir el costo computacional en este algoritmo para así no sólo conseguir un camino óptimo sino también que el proceso sea corto y fácil de realizar.

No se adecua si el entorno cambia, solo da una respuesta del entorno que conoce.

No se puede realizar el proceso de manera paralela.

D*

Encuentra siempre la solución para el problema planteado.

Los pasos están definidos para poder cumplir con su objetivo.

Utiliza una teoría similar a la planteada en el algoritmo A* para realizar el proceso.

Es un algoritmo finito ya que encuentra la solución sin probabilidad de que se desborde.

D* siempre encuentra la mejor solución para la trayectoria entre dos puntos.

Tiene en cuenta obstáculos y costes para que el camino sea el adecuado.

El tiempo de ejecución es parecido al algoritmo de A* ya que las características del proceso son muy semejantes.

El costo computacional es medio, aunque debe estar recalculando el espacio de trabajo y por esa razón es mayor que el costo computacional del anterior algoritmo.

Este algoritmo está diseñado para poder adecuarse si el entorno cambia, es la mayor diferencia (mejora) con el algoritmo A*.

No se puede realizar el proceso de manera paralela.

D* Lite

Siempre obtiene la solución para el problema planteado.

Los pasos están definidos para poder cumplir con su objetivo.

Utiliza un proceso similar al algoritmo D* para alcanzar la meta.

Es un algoritmo finito ya que encuentra la solución sin probabilidad de que se desborde.

D* lite encuentra la mejor solución para la trayectoria entre dos puntos según como se vaya modificando el entorno.

Tiene en cuenta obstáculos y costes para que el camino sea el adecuado.

El tiempo de ejecución es parecido al algoritmo de A* ya que las características del proceso son muy semejantes.

El costo computacional es medio, aunque debe estar recalculando el espacio de trabajo.

Este algoritmo está diseñado para adecuarse al cambio del entorno.

No se puede realizar el proceso de manera paralela.

Dynamic Search

Obtiene una solución concreta para el problema planteado.

Los pasos están definidos para poder cumplir con su objetivo.

Utiliza un proceso similar al algoritmo de *Dijkstra* para alcanzar la meta.

Es un algoritmo finito ya que encuentra la solución sin probabilidad de que se desborde.

Encuentra la mejor solución para la trayectoria entre dos puntos según como se vaya modificando el entorno.

Tiene en cuenta obstáculos y costes para que el camino sea el adecuado.

El tiempo de ejecución depende de la modificación que se encuentre en el entorno.

El costo computacional es bajo si se ejecuta en GPU

Este algoritmo funciona en entornos dinámicos.

Este algoritmo puede realizar el proceso de manera paralela.

En las tablas 5 y 6 se representa si se cumplen las características generales de los algoritmos estudiados y en qué nivel refiriéndose a una puntuación según el desempeño.

Tabla 4

Esquema del Cumplimiento y Valoración de las Características Generales de Algoritmos de Planificación de Trayectorias para Entornos Estáticos

Características generales de los Algoritmos	**Wanderplanner**	**Spanning tree**	**Breadth first**	**Dijkstra**	**A***
Exactos			X	X	X
Sistemáticos	X	X	X	X	X
concretos			X	X	X
Finitos			X	X	X
Admisibles			X	X	X
Factible	X	X	X	X	X
Eficiencia respecto al tiempo	*	**	***	****	** ***
Costo computacional	*	**	***	****	** ***

Fuente. Elaboración propia del documento (2023).

Tabla 5

Esquema del Cumplimiento y Valoración de las Características Generales de Algoritmos de Planificación de Trayectorias para Entornos Dinámicos

Características generales de los Algoritmos	**D***	**D* lite**	**Dynamic search**
Exactos	X	X	X
Sistemáticos	X	X	X
concretos	X	X	X
Finitos	X	X	X
Admisibles	X	X	X
Factible	X	X	X
Eficiencia respecto al tiempo	******	******	******
Costo computacional	******	******	******
Tiempo real	X	X	X
Paralelizable			X

Fuente. Elaboración propia del documento (2023).

Nota: En la eficiencia respecto al tiempo y el costo computacional, los asteriscos indican una calificación, donde a mayor número de asteriscos mejor desempeño tendrá en esa característica**.**

Requerimientos de la Aplicación

En este paso dentro del método propuesto se desea diseñar un procedimiento para la evaluación de algoritmos de planificación de trayectorias para personas invidentes, utilizando criterios de optimización de tiempo y espacio. Se analiza el caso particular: cómo se movilizan las personas invidentes y cómo se puede ayudar en el desplazamiento.

Análisis del Caso Particular: Cómo se Movilizan las Personas Invidentes

El bastón blanco es la herramienta más utilizada por los invidentes para detectar obstáculos a corta distancia. Cuando la persona invidente conoce los caminos para llegar a un lugar significa que ha hecho un mapa en su cerebro que hace que sienta seguridad para desplazarse solo. Si no hay reconocimiento del entorno por donde se desea movilizar es muy posible que se encuentre en riesgo de perderse del camino o de sufrir un accidente. Si la persona con dificultad severa de visión transita por entornos desconocidos necesita de ayuda para desplazarse.

Al analizar el entorno donde se pueden movilizar las personas invidentes tenemos que puede ser entornos dinámicos y estáticos, también interiores y exteriores.

Los entornos estáticos son aquellos donde se tiene un conocimiento completo de él y su mapa no cambia en el tiempo.

Los entornos dinámicos son aquellos de los cuales solo se tiene una información parcial y los obstáculos u objetos dentro de él pueden cambiar su posición en el tiempo. (Maikel, 2009).

Los entornos interiores permiten un conocimiento de los objetos que comúnmente no se modifican de lugar, permitiendo que la persona con discapacidad se familiarice y no choque con ellos, logrando una adaptación completa si conoce el lugar, pero si no lo conoce siempre estaría en riesgo y requiere una ayuda adicional para poder desplazarse con tranquilidad. Existen dispositivos que han permitido que personas ciegas se puedan movilizar libremente en entornos interiores, por ejemplo: CCNY *Smart Cane* (Q. Chen et al.,2017) El dispositivo móvil se utiliza para reconocer una ubicación en interiores y proporcionar una guía de navegación a través de audio al usuario a un destino específico. Utiliza un algoritmo de planificación de trayectorias

llamado A* (Kelly, 2017) trabaja en entornos interiores y guía con eficiencia al invidente a su objetivo.

Cuando se trata de entornos exteriores es muy difícil que el invidente se sienta seguro al desplazarse. Esto lleva a que la persona se sienta incómoda al salir porque debe hacerlo acompañado, solo se expone a sufrir un accidente. Al restringir su movimiento baja su calidad de vida, por lo que para mejorarla se han creado dispositivos con sistemas de detección de obstáculos para invidentes, como *Wearable RGBD Indoor Navigation System for the Blind*. (Li, 2016) utiliza el algoritmo D* (Koenig, 2002) "que realiza planificación de rutas en tiempo real.

Existen aplicaciones de planificación de trayectorias en diferentes situaciones que ayudan a cualquier persona, sin necesidad de tener un tipo de discapacidad.

El movimiento de una aeronave en un determinado espacio siguiendo una secuencia de puntos de ruta, es un gran ejemplo, la implementación de algoritmos que generen trayectorias definidas un punto inicial y final, que garanticen tanto la evitación de obstáculos como la optimización de dichas trayectorias según diversos objetivos (JD, 2015), es un estudio muy común que ha generado magníficos resultados.

Google Maps está desarrollado para encontrar, ubicar y marcar lugares, encontrar direcciones, duraciones y muchas otras tareas basadas en Sistemas de Información Geográfica (SIG). El Sistema de Información Geográfica es un Sistema de apoyo a la toma de decisiones que tiene datos geoespaciales (por ejemplo, mapas) para analizar los datos espaciales y las relaciones entre ellos (Brian, 1996). *Google Maps* cuenta con un potente algoritmo que se encarga de hacer todos los cálculos necesarios para encontrar la ruta más corta a nuestro destino, adivinar el tráfico y establecer rutas. Para ello, ha desarrollado estas funciones gracias al recurso de la Inteligencia

artificial (IA) con la cual se analiza patrones históricos de tráfico para las carreteras en el tiempo (RRP 2020).

Cómo se Puede Ayudar en el Desplazamiento.

Existen ayudas tecnológicas que orientan y hacen que la navegación por las calles de las ciudades guíe a personas con discapacidad visual, logrando una mejora importante en su calidad de vida y su autonomía de movimientos. Hay asistentes para desplazamiento en entornos dinámicos que se vienen estudiando, mostrando excelentes resultados que además de detectar obstáculos determinan rutas seguras.

Ashirase (Cuevas, 2021) es un sistema de navegación que consta de una aplicación de teléfono inteligente y un dispositivo de vibración tridimensional que incluye un sensor de movimiento, que se adjunta dentro del zapato. Según la ruta establecida con la aplicación, el dispositivo vibra para proporcionar navegación. Cuando el usuario debe ir derecho, el vibrador ubicado en la parte delantera del pie vibra, y cuando el usuario se acerca a un giro a la derecha o izquierda, el vibrador del lado derecho o izquierdo vibra para notificar al usuario.

La empresa de Baja Austria *Tec-Innovation* ha desarrollado un zapato inteligente para detectar obstáculos. El zapato, conocido como *InnoMake*, se ha comercializado recientemente como un dispositivo médico aprobado y está destinado a hacer que la movilidad de las personas ciegas y con discapacidad visual sea más segura. Este sistema detecta obstáculos, pero no determina rutas óptimas (Eigner S. 2021).

La realimentación es una característica muy importante para tener en cuenta al momento de identificar sistemas tecnológicos que ayudan al desplazamiento de invidentes. Debe ser inmediata que se refiere a que si existe cambio del entorno pueda replantear y ajustar la ruta a

una segura. Esto se logra trabajando algoritmos que se desempeñen en entornos dinámicos. También es conveniente analizar que el invidente no cuenta con un sentido que es el de la visión que hace que su oído sea su apoyo, por lo que al sobrepasar su capacidad cognitiva con realimentación auditiva se limitaría su reacción. Los patrones de vibración son comúnmente utilizados en dispositivos para invidentes porque apoyan su desplazamiento avisando a través de sensores vibratorios donde puede desplazarse.

Selección de Algoritmos de Planificación de Trayectorias

Según las Tablas 4 y 5 que analiza el comportamiento de algoritmos de planificación de trayectorias estudiados, se escogerán los algoritmos que logren un orden desde el más fácil al más complejo y eficiente. Es importante resaltar que iremos analizando desde el algoritmo de menor complejidad en su implementación, realizando mejoras y cambios en el código para un avance paulatino hasta llegar al algoritmo que realice el proceso que nos permitirá escoger el algoritmo de planificación de trayectorias más eficiente.

El objetivo inicial del proyecto presentado por Díaz et al., (2019). “Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real” era crear un dispositivo que apoyara el movimiento seguro, en un entorno desconocido, de una persona con problemas de visión. Al lograr este objetivo se quiere incorporar un módulo de planificación de trayectorias, y así implementar el algoritmo más adecuado en el sistema y poder garantizar que las nuevas tecnologías lleguen a las personas que padecen cada día por su discapacidad.

Después de estudiar diferentes recursos bibliográficos se opta por seguir la secuencia tomada en el libro *Motion Planing,* (Kelly, 2017) donde encontramos una guía muy acertada, porque además de sugerir los algoritmos, se cuenta con las instrucciones para la implementación,

mostrando un pseudocódigo. Esto permite que al momento de implementar cada línea de código tenga un significado claro en su resultado.

Realizaremos la comparación de 5 algoritmos para entornos estáticos y 2 algoritmos para entornos dinámicos. La secuencia empleada en el capítulo 10 del libro *Motion Planing*, nos guía perfectamente a determinar que se puede ir de un algoritmo sencillo, pasando a mejorar la efectividad implementando algoritmos con mayor dificultad, llegando a la conclusión que, al aumentar la complejidad, de manera sistemática se llega a resultados óptimos.

Selección de Lenguaje de Programación y Herramientas de Desarrollo

Se analizarán diferentes leguajes de programación para poder estudiar las ventajas y desventajas de cada uno y así seleccionar el más indicado para este proceso.

MATLAB

Es un entorno optimizado para realizar diseños, exploraciones y soluciones de un problema de una manera más interactiva haciendo uso de un lenguaje de alto nivel.

Su interfaz otorga muchas facilidades que brindan una ayuda al usuario para visualizar diferentes características de su trabajo lo cual hace mucho más sencillo la corrección y la evaluación del rendimiento de los códigos que se estén trabajando.

Matlab posee un alto número de herramientas para aplicaciones en el campo científico y de ingeniería para trabajar en muchos ámbitos o diferentes temáticas esto lo hace un software muy útil y uno de los más utilizados en muchas áreas de la investigación. También permite hacer una interacción o comunicación con otros lenguajes de programación o con diferentes dispositivos programables del exterior para realizar tareas como recibir, enviar o procesar datos lo cual es muy necesario cuando el microprocesador externo que se posee no tiene toda la capacidad para realizar estas tareas. (*The MathWorks*,2022).

C++ y C

El lenguaje C fue uno de los primeros que lograron alcanzar un alto nivel de renombre ya que permite realizar o desarrollar aplicaciones en cualquier campo con un rendimiento grande. Al paso del tiempo, con la nueva evolución en el campo de la programación, este lenguaje presentó la necesidad de ser actualizado, para poder realizar mejores aplicaciones y por ello nació C++.

El lenguaje de programación C++ le añade a su antecesor C la posibilidad de trabajar y manipular objetos, lo que lo convirtió en unos de los más interesantes en el mercado debido a su alto nivel. En la actualidad C++ continúa estando en consideración de los programadores como opción para el desarrollo de proyectos en los cuales se necesita conseguir buenos resultados con respecto a la velocidad de ejecución y el rendimiento del software (Hosting Plus, 2022).

JAVA.

Es un lenguaje de programación de alto nivel derivado de C y C++ con la diferencia que no posee las características menos usadas y más confusas de los anteriores lenguajes, lo que lo hace más sencillo.

Posee un enfoque orientado a objetos que es uno de los estilos de programación más utilizados. También permite realizar un diseño de software donde los distintos tipos de datos que se estén usando se encuentren unidos a sus correspondientes operaciones.

Este lenguaje tiene una gran biblioteca y muchas herramientas para crear y distribuir software, además, puede ejecutarse en cualquier hardware, lo que hace muy accesible y portátil (Carranza, 2021).

PYTHON

Es utilizado para el desarrollo de aplicaciones y programas en muchos ámbitos. Es un lenguaje de programación de alto nivel que es interpretado, lo que quiere decir que no es

necesario que su código sea compilado para ser ejecutado, sólo se ejecuta directamente haciendo uso de un programa intérprete para esto.

Es uno de los lenguajes más utilizados actualmente por su sencillez, debido a que es un lenguaje gratuito, su capacidad multiforme y su facilidad para el desarrollo de aplicaciones con tecnologías modernas como la inteligencia artificial (Hosting Plus,2021)

Se utiliza para:

Programación de servidores.

Análisis de grandes volúmenes de datos (*big data y business intelligence*) (Rendon,2021).

Desarrollos de software de inteligencia artificial (IA) (Pascual,2021).

CUDA

Debido al alto potencial que tiene la GPU (*Graphics Processing Unit*) para ayudar a las investigaciones y a la comunidad científica en general, NVIDIA realizó una modificación a sus GPUs para adaptarlas y que fueran programables y además añadir soporte para otros lenguajes de programación como C y C++.

Se introdujo la arquitectura CUDATM (*Compute Unified Device Architecture*), esta arquitectura permite un cálculo en paralelo de propósito general, con una nueva colección de instrucciones y un nuevo modelo de programación paralela, que posee un soporte para lenguajes de alto nivel, que son constituidas por cientos de núcleos que pueden procesar de manera concurrente miles de hilos de ejecución. En este tipo de arquitectura cada núcleo tiene ciertos recursos compartidos entre los cuales se puede encontrar los registros y memoria. (Represa, 2016).

El lenguaje de programación que se escogió para la implementación es Matlab, porque tiene muchas ventajas ante los demás. Este lenguaje permite expresar directamente arreglos

vectoriales y matrices matemáticas, las cuales facilitan mucho el desarrollo de este proyecto. Matlab está diseñado para la ciencia y la ingeniería por lo que no necesariamente se debe ser ingeniero informático para poder entender su contenido, sus funciones, líneas de código y todas sus herramientas, por esto su entorno de trabajo es muy amigable, fácil de manejar y entender. Todas las librerías que proporciona Matlab están optimizadas para mayor eficiencia en la ejecución de los códigos. Los *toolboxes* o cajas de herramientas de Matlab ofrecen funcionalidades desarrolladas profesionalmente, probadas rigurosa y totalmente documentada para una amplia gama de aplicaciones científicas y de ingeniería. Los *toolboxes* están diseñados para trabajar en conjunto, y se integran con entornos de computación paralela, GPUs y generación de código C.

Implementación de Algoritmos de Planificación de Trayectorias

Propuesta Pseudocódigo General.

Un pseudocódigo es una representación informal de un algoritmo, donde se utiliza la organización de un lenguaje de programación normal, pero cualquier persona puede leer en esta forma. El fin del pseudocódigo presentado en la Figura 13 es el de simbolizar la solución de un algoritmo de la forma más detallada posible, y a su vez lo más parecido al lenguaje que después se utilizará para la programación de este. En la página 13 se encuentra la Tabla 1 Siglas o acrónimos, ahí se presentan cada una de las siglas con su respectivo significado.

Figura 13

Pseudocódigo General

```
00    Algoritmo de planificación de trayectorias (Xs, Xg)
01    X ← X_s
02    O. Inserta (X)
03    while (O≠Ø)
04          x ← O.remove()
05          C.inser(x)
06          if(x = x_g) return success
07          expandNode()
08    endwhile
09    return failure
```

Fuente. Modificado de Kelly (2017, p.640-690).

El proceso general para realizar la implementación pide tener un punto inicial y un destino. Para el análisis utiliza dos listas. Se inicia introduciendo el punto inicial a una de las listas denominada *OPEN* (Lista abierta) y se comienza un bucle, donde existen tres funciones fundamentales. En la primera función encontramos el proceso de remover un elemento de la lista *OPEN*. Este elemento removido ya dependerá de cada algoritmo en particular. La segunda, hace referencia a que se inserta dicho elemento, removido anteriormente, en la siguiente lista la cual se la conoce como *CLOSE* (Lista cerrada). En la última función se expande el nodo, y se evalúa los nodos vecinos al nodo actual. Estos serán ajustados según especifique de cada algoritmo y se ingresará los vecinos en la lista *OPEN*. El procedimiento se repetirá cíclicamente hasta que el punto actual que se va a evaluar, que es el elemento removido de *OPEN*, sea igual al destino o punto final o en el caso de que la lista *OPEN* se encuentre vacía.

Pasos para la Implementación

Como regla principal los algoritmos de planificación de trayectorias que se trabajarán deben basarse en el pseudocódigo general.

El primer paso consiste en definir el espacio de trabajo. Se encontrarán las características principales como la cantidad de nodos que tiene el grafo, los obstáculos y, como lo sugiere el pseudocódigo, definir punto inicial y punto final. Al seguir el paso a paso del algoritmo general se puede construir los códigos de los algoritmos que se escogieron, pero hay que tener en cuenta las condiciones que definen cada uno de ellos.

Se inicia ingresando el punto inicial a la lista *OPEN*.

Continua un ciclo *while*, el cual tiene como característica fundamental que su condición siempre permitirá el ingreso al proceso interno del ciclo. Esta condición no depende del número de iteraciones que debería realiza el *while*, debido a que no se puede predefinir la cantidad de veces que se repetirá. Se debe tener en cuenta otros aspectos que determinen cuando finalice el ciclo: cuando la lista *open* esté vacía (O≠Ø) o se llegue al punto final (*x = xg)*.

El paso siguiente es crear las tres funciones principales que menciona el pseudocódigo. Cabe aclarar que cada código tendrá sus propias especificaciones y modificaciones a estas funciones, para cumplir con la tarea correspondiente.

La primera función denominada *O.Remove* (remover de *OPEN*) consiste en tomar un dato que se encuentra en la lista *OPEN* y removerlo o quitarlo. El dato que va a ser removido dependerá de lo que cada algoritmo en particular sugiera.

La segunda función se denomina *C.inser* (insertar en *CLOSE*) y tiene la tarea de introducir en la lista *CLOSE* el nodo que fue removido en el paso anterior. En este caso la forma en que debe ser ingresado no tiene inconveniente ya que el orden en el que sean ubicados en la lista *CLOSE* no alterará los resultados.

Como tercera función se encuentra *EXPANDNODE* (expandir el nodo) y en general evalúa los nodos vecinos al actual. Cada algoritmo especificará qué datos se deben calcular para

cada vecino y después serán introducidos en la lista OPEN en el orden como lo indique dicho algoritmo.

El condicional nos ayudará a identificar si se ha completado la tarea y eso se logra identificar cuando el punto o nodo actual sea igual al nodo de destino, en este momento se debe parar el ciclo y finalizar el proceso. Este proceso en el pseudocódigo general aparece en medio de la función 2 y la función 3, pero en algunos algoritmos este condicional va dentro de la función 3, igualmente tiene la misma tarea.

Al terminar el ciclo la lista *CLOSE* dará los datos correspondientes para obtener la ruta donde el proceso consiste en tomar el punto final y retroceder con el padre de cada nodo y continuar el proceso hasta llegar al punto inicial y así obtener el camino o trayectoria, es decir, el camino se determina desde el último nodo hasta el primero.

Equipo de Computo

Computador portátil Asus VivoBook S14.

Características

Nombre del dispositivo:DESKTOP-DJK7Q6V

Procesador: Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

RAM instalada: 4,00 GB (3,88 GB usable)

Identificador de dispositivo: 2EEBE28E-7B7F-4172-A404-462F5A1256CD

Id. del producto: 00331-90000-00001-AA795

Tipo de sistema: Sistema operativo de 64 bits, procesador basado en x64

Windows 10.

Selección de Criterios de Evaluación de Desempeño de los Algoritmos Implementados.

Se deben realizar pruebas que permitan comparar el desempeño de los algoritmos a implementar, luego hacer comparaciones con respecto a estos criterios:

Tiempo de ejecución: Este parámetro hace referencia a la duración, en unidades de tiempo, que tardó el algoritmo en calcular la ruta hasta llegar al objetivo. También con esta medición se puede determinar si el costo computacional fue elevado o no, con lo anterior se tiene un factor muy importante para la comparación de los algoritmos en cuanto a la eficiencia.

Celdas expandidas: Cada algoritmo sigue las condiciones de acuerdo con sus características. Este proceso analiza todas las celdas que serán expandidas, las cuales arrojarán datos de valor que sirve como parámetro comparable. A mayor número de celdas expandidas, mayor será el costo computacional

Número de celdas del camino: Son las celdas que fueron escogidas para formar el camino. En la eficiencia de los algoritmos lo que se busca es que el camino sea lo más directo posible hacia la menta, en otras palabras, un camino óptimo será el que posea el menor número de celdas según marque las condiciones del espacio de trabajo.

Número de iteraciones: Las iteraciones hace referencia a los pasos que el algoritmo ejecutó para cumplir su tarea. Mientras menos iteraciones posea quiere decir que será menor costo computacional.

Efectividad al llegar al objetivo: En la evaluación de estos algoritmos este parámetro es considerado muy importante debido a que la función principal es verificar si existe una ruta que llegue al destino y dado el caso que pueda calcularla o en el caso contrario indicar que no existe camino alguno. Existen algunos algoritmos que no lograrán esta meta, lo que hará que esta

efectividad sea menor y esto concluirá que no es confiable o que no logra dar una respuesta clara al problema planteado.

Evaluación de Desempeño

Ahora se valida el método de evaluación de algoritmos en el caso práctico planteado.

Se mostrará los pasos de la implementación, inicialmente un pseudocódigo de cada uno de los 5 algoritmos implementados, y la explicación de cómo funciona. Se realizan pruebas iniciales tomando grillas de ocupación en un punto de inicio y otro destino. Se comparan algoritmos en entornos estáticos con grillas de 100x100, 200x200 y 400x400 casillas. En cada tamaño se realizó tres pruebas, cambiando el porcentaje de obstáculos (20%, 40%, 50%). Se obtienen los resultados y se realiza su comparación según los criterios de evaluación acordados. En entornos estáticos se implementan 5 algoritmos: *Wander Planner, Spanning tree, Breadfirst, Dijkatra y A start*. En entornos dinámicos se comparan 2 algoritmos D start y *A star From Scratch*.

Wander Planner.

Es el algoritmo planificador más básico, y en la literatura (Kelly 2017), se encuentra literalmente que no es un planificador debido a su estructura que no es similar a la presentada en el pseudocódigo general de la figura 13, dado que porque no maneja lista *OPEN* ni lista *CLOSE*. Su proceso fundamental es escoger de manera aleatoria uno de los vecinos que tenga el nodo actual, con la única condición de que aquel nodo escogido no sea un obstáculo.

Figura 14

Pseudocódigo para el Algoritmo Wander Planner

```
00   algorithm wanderPlanner()
01   x ← x_s
02   while(true)
03       for some(u ϵ U(x))
04           x ← f(x,u)
05           If(x = x_g) return success
06       endfor
07   endwhile
08   return
```

Fuente. Tomado de Kelly (2017, pp.640-690).

Wander Planner realiza esta tarea hasta que se encuentre el punto final o *goal*. No es completamente seguro que pase, debido a la aleatoriedad con la que realiza el proceso o del tiempo de ejecución que es bastante elevado.

Se puede observar en la Figura 14 que el proceso en el pseudocódigo es muy sencillo. Consiste en seleccionar punto actual y escoger uno de sus vecinos al azar. Éste se convierte en el nuevo punto actual y continua el proceso hasta que la línea 5 del pseudocódigo sea verdadera y en ese momento termina el proceso.

Como resultados en este algoritmo se obtuvieron rutas muy complicadas y nada optimas. También el tiempo de ejecución es bastante elevado dado que se presentan muchas rutas cíclicas.

Función ExpandNode.

Esta función es utilizada por los siguientes algoritmos y es importante entender su tarea (ver el pseudocódigo en la Figura 15). Se aclara que hay ciertas variaciones de esta función para cada algoritmo. Se encarga de evaluar los nodos vecinos que tiene el nodo actual, a los cuales le calcula los costos y con ello determina en qué lista y en qué orden irá.

Figura 15

Pseudocódigo para la Función ExpandNode

```
00    algorithm expandNode(x)
01    for each(u ϵ U(x)
02        x_next ← f(x, u))
03        if(x_next = x_g)
04              x_next · parent ← x;return success
05        else if(x_next ∉ O && x_next ∉ C)
06              x_next · parent ← x
07              O.insert(x_next)
08        endif
09    endfor
10    return failure
```

Fuente. Tomado de Kelly (2017, pp.640-690).

Se comienza evaluando uno por uno cada vecino factible que tenga el nodo actual. A cada uno se le calcula los datos pertinentes según lo pida el algoritmo. Después se realizan comparaciones haciendo uso de condicionales y se toma la decisión en que lista irá el nodo y en qué posición de la lista debe ir.

Spanning tree.

Al realizar algunas mejoras al algoritmo *Wander Planner*, se obtiene *Spanning tree*. Éste utiliza un concepto mucho más amplio para realizar el análisis y planificación. Utiliza el árbol de expansión para que en el proceso se pueda tener en cuenta los nodos que ya se utilizaron evitando rutas cíclicas.

En el pseudocódigo de la Figura 16 se puede encontrar: la lista *OPEN*, donde se posicionan los vecinos que se planean visitar y la lista *CLOSE*, en donde están los nodos que ya han sido visitados o ya fueron nodos actuales, en las cuales se agregan los datos correspondientemente. Con estas listas se logra escoger las siguientes posiciones de manera

aleatoria, pero ahora con dos condiciones: que el nodo no sea un obstáculo y que tampoco sea uno que ya se haya evaluado, para garantizar que no se lo vuelva a evaluar ni a posicionar.

Figura 16

Pseudocódigo para el Algoritmo Spanning tree

```
00    algorithm spanning Tree(x_s, x_g)
01    O.insert(x_s)
02    x_s · parent ← null
03        while(O ≠ Ø)
04            x ← O.remove()
05            C.inser(x)
06            if(expandNodeST(x)) return success
07        endwhile
```

Fuente. Tomado de Kelly (2017, pp.640-690).

Como resultados al realizar las pruebas de este algoritmo se nota un gran avance o mejores resultados que su antecesor, el algoritmo *Wander Planner*, ya que las trayectorias no son cíclicas, sino que avanzan hacia la meta sin repetir ningún nodo. Todo esto se puede observar en la cantidad de casillas que posee el camino y además el tiempo de ejecución disminuye de manera considerable. También se tiene un resultado negativo: hay casos en los que el camino se encierra con el mismo, mientras va haciendo el recorrido, esto hace que no siempre encuentre la trayectoria hacia la meta.

Breadthfirst.

Desde este punto los algoritmos comienzan a utilizar cálculos más avanzados para buscar un resultado mucho óptimo y siempre se realiza el análisis antes de tomar la decisión o de elegir el camino. Todo esto hace que el proceso sea un poco más complejo ya que el algoritmo tendrá que realizar muchos más pasos que los anteriores algoritmos *Wander Planner* y *Spanning tree*. En la Figura 17 se muestra el pseudocódigo para el algoritmo *Breadthfirst*.

Figura 17

Pseudocódigo para el Algoritmo Breadthfirst

```
00    algorithm breadthFirst(x_s, x_g)
01    O.insertLast(x_s)
02    x_s · parent ← null
03        while(O ≠ Ø)
04            x ← O.removeFirst()
05            C.inser(x)
06            if(expandNodeBF(x)) return success
07        endwhile
08    return failure
```

Fuente. Tomado de Kelly (2017, pp.640-690).

El proceso de este algoritmo es examinar los nodos del árbol de expansión colocando el respectivo padre. Cada nodo que se va a evaluar se inserta al final de la lista *OPEN* y la asignación del nodo actual se hace con el que queda de primero en esta lista el cual es insertado en la lista cerrada. Esto hace que el algoritmo evalúe por niveles. Al encontrar el punto final se realiza la búsqueda del camino iniciando siempre desde el punto final y devolviéndose por los padres de cada uno, haciendo que se encuentre la trayectoria que conecta el nodo *start* con el *goal*.

Los resultados de este algoritmo son buenos ya que en primer lugar puede determinar si existe un camino o no y de existir garantiza que lo encuentra y siempre va a ser el camino óptimo lo que no se lograba con los algoritmos anteriores. Las desventajas que posee, según los resultados que se obtuvieron, es que el número de casillas expandidas o analizadas es siempre muy elevado y esto conlleva a que el tiempo de ejecución sea mayor para ciertos casos.

Dijkstra.

Este algoritmo se parece mucho al algoritmo *Breadth first* ya que utiliza el mismo proceso, como se ve en el pseudocódigo de la Figura 18, con la diferencia que para ordenar los

nodos en las listas se basa, no en el nivel, sino en un nuevo parámetro denominado ganancia. Esta ganancia hace referencia a la dificultad que se presenta al moverse de un nodo al otro, que es de dos tipos: la ganancia de borde y la ganancia de terreno, esta última no se la tendrá en cuenta para el estudio y la comparación de los algoritmos.

Figura 18

Pseudocódigo para el Algoritmo Dijkstra

```
00    algorithm Dijkstra(x_s, x_g)
01    x_s.g ← 0
02    O.insertSorted(x_s, x_s.g)
03    x_s · parent ← null
04        while(O ≠ Ø)
05            x ← O.removeFirst()
06            C.inser(x)
07            if(x = x_g) return success
08            expandNodeDijkstra(x)
09        endwhile
10    return failure
```

Fuente. Tomado de Kelly (2017, pp.640-690).

Como podemos observar el pseudocódigo de la Figura 18 es el mismo que el de *Breadth first,* con la diferencia que en la línea 2 y en la función *expandNodeDijkstra* de la línea 8, se insertan los nodos considerando el costo g. La diferencia entre *Breadfirst y Dijkatra* se la encuentra dentro de la función de "*ExpandNode*" en la cual, en primer lugar, realiza el cálculo de g, que consiste en sumar el costo de borde más el coste que tiene el padre del nodo que está siendo evaluado. El costo de borde se lo toma con dos valores. Si el movimiento es axial posee un valor determinado, de lo contrario si el movimiento es diagonal el valor es el resultado de la raíz cuadrada del doble del valor axial al cuadrado.

Teniendo el valor de la ganancia, la función *insertSorted* se encarga de acomodar los nodos en la lista *OPEN* en orden, colocando siempre de primero el que tenga menor valor de g,

esto para que siempre escoja el más cercano al nodo actual. Todo este proceso se realiza para disminuir la cantidad de casillas evaluadas.

Después de realizar las pruebas de este algoritmo se observa que obtiene el mismo camino óptimo que su antecesor, pero ahora la cantidad de casillas expandidas y evaluadas es un poco menor por lo cual también disminuye el tiempo de ejecución.

A star

Es un algoritmo que tiene un proceso muy semejante a los dos anteriores algoritmos (se observa en la Figura 19), BFS y DIJ, pero se ayuda de tres términos muy importantes, la ganancia(g), que se refiere a la ganancia del nodo inicial al nodo actual, la heurística de Manhattan (H), que calcula cuántas casillas hay del nodo actual al nodo final y la estimación (F), que es la suma de g y de H, correspondiente el costo total. F prioriza qué nodos son más importantes para ubicarlos en las posiciones iniciales de la lista *OPEN* y dirigir la búsqueda hacia el nodo final y con ello disminuir la cantidad de casillas evaluadas de las que tenía el anterior algoritmo.

Figura 19

Pseudocódigo para el Algoritmo Astar

```
00   algorithm Astar(x_s, x_g)
01   x_s.g ← 0; x_s.f ← calcF(x_s, 0)
02   O.insertSorted(x_s)
03   x_s · parent ← null
04      while(O ≠ Ø)
05          x ← O.removeFirst()
06          C.inser(x)
07          if(x = x_g) return success
08          expandNodeAstar(x)
09      endwhile
10   return failure
```

Fuente. Tomado de Kelly (2017, pp.640-690).

El procedimiento, como lo podemos ver en el pseudocódigo de la Figura 19, es igual que los dos anteriores algoritmos *Breadfirst y Dijkatra* la principal diferencia está en la forma de insertar los nodos en la lista *OPEN*, ya que para este proceso tiene en cuenta no sólo la ganancia sino la estimación. Teniendo esto se utilizan condicionales para escoger la lista y la posición en donde debe ir el nodo, pero ahora el valor que se utiliza para esto es la estimación F: los nodos que posean menor estimación F se ubicarán más adelante en la lista OPEN, se realiza un cálculo y una expansión de los nodos, pero redirigida hacia el punto final.

Después de haber realizado las pruebas se obtuvo que éste obtiene el mismo camino óptimo que se tiene desde el algoritmo *Breadth First* pero éste expande y evalúa muchas menos casillas y esto a su vez hace que el algoritmo reduzca el tiempo de ejecución.

D star

Este algoritmo está diseñado para entornos dinámicos. Los caminos pueden sufrir cambios mientras se está en el proceso o se recorre por caminos ya calculados. El proceso es mucho más complicado que los cinco anteriores algoritmos y su complejidad se la puede observar claramente en el Pseudocódigo de la Figura 20.

Figura 20

Pseudocódigo para el Algoritmo Dstar

```
00  algorithm initializeDstar(x_s, x_g)
01  x_s.rhs ← 0
02  O.inserSorted(x_s, calcKey(x_s))
03  x_s.parent ← null
04  computeOptimalPath()
      01  while(f[O.peek()] < f(x_g)  or  x_g.g ≠ x_g.rhs)
      02      x ← O.removeFirst()
      03      expandNodeDstar(x)
                01  if(x.g > x.rhs)
                02      x.g ← x.rhs
                03      for each(u ∈ U(x))
                04          x_next ← f(x, u)
                05          updateVertex(x_next)
                06      endfor
                07  else
                08      x.g ← ∞
                09      for each(u ∈ U(x))
                10          x_next ← f(x, u)
                11          updateVertex(x_next)
                12      endfor
                13      updateVertex(x_next)
                14  endif
                15 return
      04  endwhile
      05  return
05  return
06  while(x_s ≠ x_g)
07      Scan graph for changed edge costs
08      if(edge cost change)
09        Replaning()
            01  Update the edge cost
            02  computeOptimalPath()
            03  return
10      else
11        x_last = x_start
12      endif
13  endwhile
```

Fuente. Modificado a partir de: Koenig y Likhachev (2002, p.4) y Kelly (2017, pp.640-690).

A diferencia de los anteriores algoritmos, D star no solo calcula la trayectoria, sino que acompaña en el recorrido y sigue evaluando si existen cambios en ese espacio de trabajo, y de haberlos, replanifica el camino según sea conveniente.

Para realizar todo el proceso el algoritmo aumenta otras variables además de las que ya se usaron en el anterior.

Todo el proceso se realiza dentro de la función "*compute optimal path*" que es la encargada de calcular la ruta optima, basándose en los algoritmos de entornos estáticos, haciendo uso de las listas, donde todos los datos se guardan en un arreglo mucho más grande para conservarlos, debido a que el éxito de este algoritmo es reutilizar los datos ya calculados para no tener que volver a realizar estos análisis.

La función "u*pdateVertex*" está basada en el proceso que realizan los anteriores algoritmos para ingresar y actualizar los datos en las listas, con la diferencia que este proceso tiene algunos pasos extra, entre los cuales se encuentra el analizar si se deben realizar modificaciones, si los datos pertenecen a cálculos posteriores o al actual. Con todo el proceso se logra obtener una ruta desde el punto inicial hasta el final, mientras tanto el algoritmo sigue evaluando si existe algún cambio. Al momento de presentarse un cambio en el espacio de trabajo comienza el proceso de replanificación, que comienza validando donde estuvo el cambio y según esto se determina qué nodos necesitan modificarse debido a que algunos de ellos estaban disponibles y ahora son obstáculos o también se puede presentar el caso contrario. Como los nodos cambian se debe actualizar la información de ellos, en especial de la descendencia de dichos nodos, puesto que no pueden quedar con la información antigua, ya que aportarían información falsa y esto haría que el algoritmo sufra confusiones, por lo que se procede a utilizar la función "*compute optimal path*" nuevamente, cada vez que se deba hacer la replanificación con lo que se obtendrá una trayectoria que dirija al punto final según el nuevo espacio de trabajo obtenido después de los cambios.

Una vez implementado y puesto a prueba el algoritmo, se pudo observar que el proceso es efectivo a lo largo de todo el recorrido hasta que se consigue llegar al punto final. Al realizar las replanificaciones es muy importante que se determine si muchos de los datos pertenecen al

cálculo anterior del cambio o posterior, ya que esto podía causar un error al intentar obtener el nuevo camino. Otro de los resultados importantes es que por la complejidad estructural que tiene este algoritmo su tiempo de ejecución es elevado a comparación de los algoritmos diseñados para entornos estáticos acomodados a entornos dinámicos, pero todo lo compensa con su gran eficiencia y efectividad para encontrar la ruta adecuada en cualquier caso presentado.

A Star From Scratch (A* desde cero)

Este algoritmo está basado en A star y está rediseñado para que cumpla la función de calcular la trayectoria, pero ahora para el caso de entornos dinámicos. El proceso que se estudió en A star ahora es una función que se utiliza para realizar el análisis cada vez que se deba calcular la ruta. Este algoritmo acompaña el recorrido y cuando detecta un cambio en el espacio de trabajo inicializa el proceso realizando una replanificación, en la cual se eliminan todos los datos y toma el último nodo donde quedó como punto inicial y ejecuta el proceso desde cero utilizando nuevamente A star para calcular el camino. El proceso está claro en la Figura 21.

Como se puede notar en el pseudocódigo de la Figura 21, es un algoritmo bastante simple a comparación del proceso que realiza D star y en la mayor parte de los casos se logra obtener un buen resultado. Este algoritmo tiene un problema cuando no existe ruta entre los dos puntos, debido a que está basado en un algoritmo diseñado para entornos estáticos, en este caso no lograría encontrar ninguna trayectoria y acaba el análisis en ese instante. En otras palabras, para que *A star from scratch* encuentre la ruta los obstáculos nunca deben impedir el paso.

Al realizar la implementación se obtuvo que este algoritmo tiene un tiempo de ejecución menor a D star y esto se debe a que el proceso que se realiza es mucho más simple y a la vez el comenzar de cero hace que no se deba evaluar datos anteriores, por esta razón es más veloz. Aunque es importante resaltar que, debido a la acción de inicializar, el número de casillas que se

expanden es elevado y a pesar de que se obtenga una trayectoria esta no proporcionará toda la información pertinente del proceso por lo cual se pierden datos importantes como la cantidad de pasos que se dieron o la ganancia total de dicha ruta

Figura 21

Pseudocódigo para el Algoritmo A Star From Scratch

```
00  algorithm Astar From Scratch(x_s, x_g)
01  Astar()
      00  algorithm Astar(x_s, x_g)
      01  x_s.g ← 0; x_s.f ← calcF(x_s, 0)
      02  O.insertSorted(x_s)
      03  x_s · parent ← null
      04  while(O ≠ Ø)
      05      x ← O.removeFirst()
      06      C.inser(x)
      07      if(x = x_g) return success
      08      expandNodeAstar(x)
      09  endwhile
      10  return failure
02  while(x_s ≠ x_g)
03      move to next node
04      update start
05      Scan graph for changed edge costs
06      if(edge cost change)
07        Astar()
            00  algorithm Astar(x_s, x_g)
            01  x_s.g ← 0; x_s.f ← calcF(x_s, 0)
            02  O.insertSorted(x_s)
            03  x_s · parent ← null
            04  while(O ≠ Ø)
            05      x ← O.removeFirst()
            06      C.inser(x)
            07      if(x = x_g) return success
            08      expandNodeAstar(x)
            09  endwhile
            10  return failure
07      endif
08  endwhile
```

Fuente. Modificado a partir de Kelly (2017, pp.640-690) y elaboración propia del documento (2023).

.

Elección del Algoritmo con Mejor Desempeño

Pruebas Iniciales Tomando Grillas de Ocupación en un Punto de Inicio y Otro Destino. Entornos Estáticos

Para las pruebas se tomaron las mismas condiciones iniciales en todos los algoritmos y con iguales espacios de trabajo. Se realizaron ejercicios de comparación a los cuales se les cambió el tamaño y el porcentaje de obstáculos con el fin de obtener datos que indiquen el comportamiento de cada algoritmo al aumentar grillas de ocupación y hacer más complejo el recorrido. En todos los casos se tomaron grillas cuadradas y el punto inicial siempre fue la casilla de la esquina superior izquierda y para el punto final se tomó la casilla inferior derecha, así se exige al máximo cada uno de los algoritmos. Se hicieron 3 variaciones al tamaño de las grillas que fueron de 100, 200 y 400 casillas por lado. En cada uno de los tamaños se realizaron tres pruebas; haciendo variación en la cantidad de obstáculos aumentando su porcentaje de acuerdo con el tamaño total de celdas en cada espacio de trabajo el 20, 40 y 50 por ciento de obstáculos para un total de 9 pruebas por cada algoritmo. En las Figuras de la 22 a la 30 se pueden observar el punto inicial y final de color rojo, los obstáculos poseen un color gris, las casillas que no se ocuparon quedan en color blanco mientras que el recorrido o trayectoria está en color verde, donde comienza con un verde claro y a medida que avanza se torna más oscuro hasta que llega al punto final. Los caminos generados siempre parten desde el punto inicial y terminarán cuando encuentre la meta. Se logró realizar la comparación del cálculo de trayectoria y se muestra los resultados cuando se cambia el espacio de trabajo. Se puede observar en las nueve comparaciones cómo genera la ruta cada algoritmo, según la lógica o los pasos que da cada uno, identificando el patrón de comportamiento.

Wander Planner

La ruta generada por este algoritmo es inexacta y se puede observar claramente en las Figuras de la 22 a la 29, sección A, cómo realiza los pasos de manera aleatoria sin importar si la casilla ya fue tomada o no. Por esto, pinta mucha parte del espacio de trabajo y no se puede detectar una secuencia clara en las tonalidades de la ruta. Debido a la incertidumbre de este algoritmo se puede deducir que el número de pasos de la trayectoria generada no depende del tamaño ni de las condiciones del espacio de trabajo. En el último ejercicio mostrado en la Figura 30 no se logró obtener una ruta debido a que la inexactitud e ineficiencia de *wander planner* hizo que tomara demasiado tiempo para obtener un resultado.

Spanning Tree

Genera una ruta mucho más eficaz, en comparación con el algoritmo *wander planner*, ya que realiza la búsqueda teniendo en cuanta las casillas que fueron visitadas para no tomarlas nuevamente. Se puede observar muy claramente en las Figuras de la 22 a la 30, sección B, cómo avanza en el camino por cada paso que da al notar el aumento de la tonalidad en el color de esta trayectoria. Aunque es mejor que *wander planner* todavía se pueden evidenciar fallas que hacen que la ruta no sea la mejor. En la trayectoria se dan muchos pasos innecesarios, esto sucede porque este algoritmo todavía escoge el paso siguiente de manera aleatoria. Otro resultado que se obtuvo es evidenciar que al ejecutar el algoritmo puede ser que se pare y no encuentre el objetivo, ya que el algoritmo no puede repetir los puntos ya visitados ni tampoco tomar nodos que se encuentren fuera de la grilla y llegar a un punto en el que no haya opciones de movimiento es probable. En ese caso no habría paso hacia la meta. Este problema aumenta cuando el porcentaje de obstáculos es mayor y también cuando el tamaño de la grilla aumenta.

Para obtener una trayectoria se tuvo que correr el algoritmo varias veces, hasta obtener un camino que lleve desde el punto de partida hasta la meta.

Breadth First

Expone una trayectoria óptima como se puede evidenciar en las Figuras de la 22 a la 30, sección C. Se obtiene un camino desde el punto de partida hasta el final. Esto se logra por el proceso de analizar toda la grilla y por último escoge el mejor camino. Al igual que el anterior algoritmo *spanning tree*, también se puede definir si no existe paso para llegar al objetivo.

Dijkstra

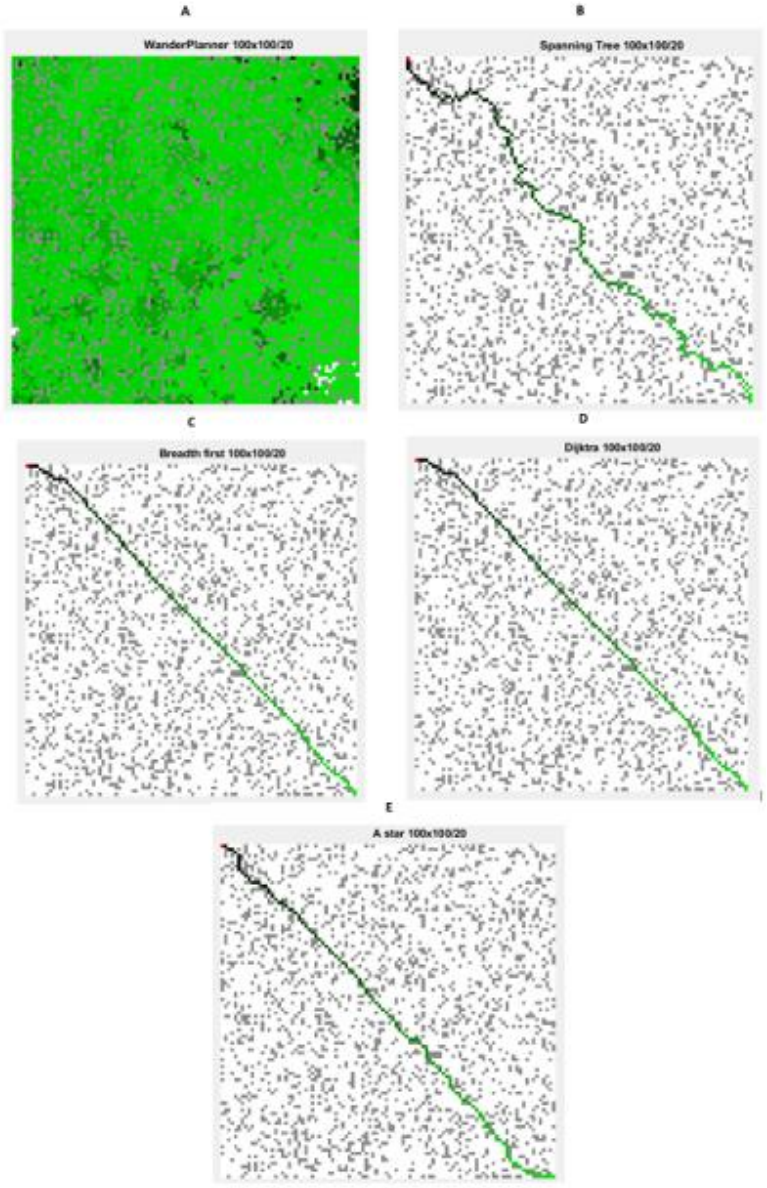
Presenta una trayectoria igual a la presentada por *Breadth First*, mostrada en las figuras de la 22 a la 30, sección D, asegurando una ruta óptima. Dijkstra hace uso del costo de borde para analizar cada nodo y así determinar la trayectoria y llegar al objetivo, dirigiendo mejor su búsqueda. Este parámetro lo ayuda a trabajar en diferentes tipos de terreno, planos y no planos, mejorando el desempeño de los anteriores algoritmos.

A star

Genera una trayectoria optima al igual que los dos algoritmos anteriores, como se observa en las Figuras de la 22 a la 30, sección E. Aunque se ve un ligero cambio en su trayectoria el camino muestra un resultado equivalente. A Star utiliza la heurística entre cada nodo y permite, junto con la ganancia, dar una estimación de la distancia entre dos nodos. Esto hace que dirija totalmente la búsqueda al objetivo. La diferencia entre los tres últimos algoritmos no está definida por la ruta que calculan sino en el cómo lo hacen.

Figura 22

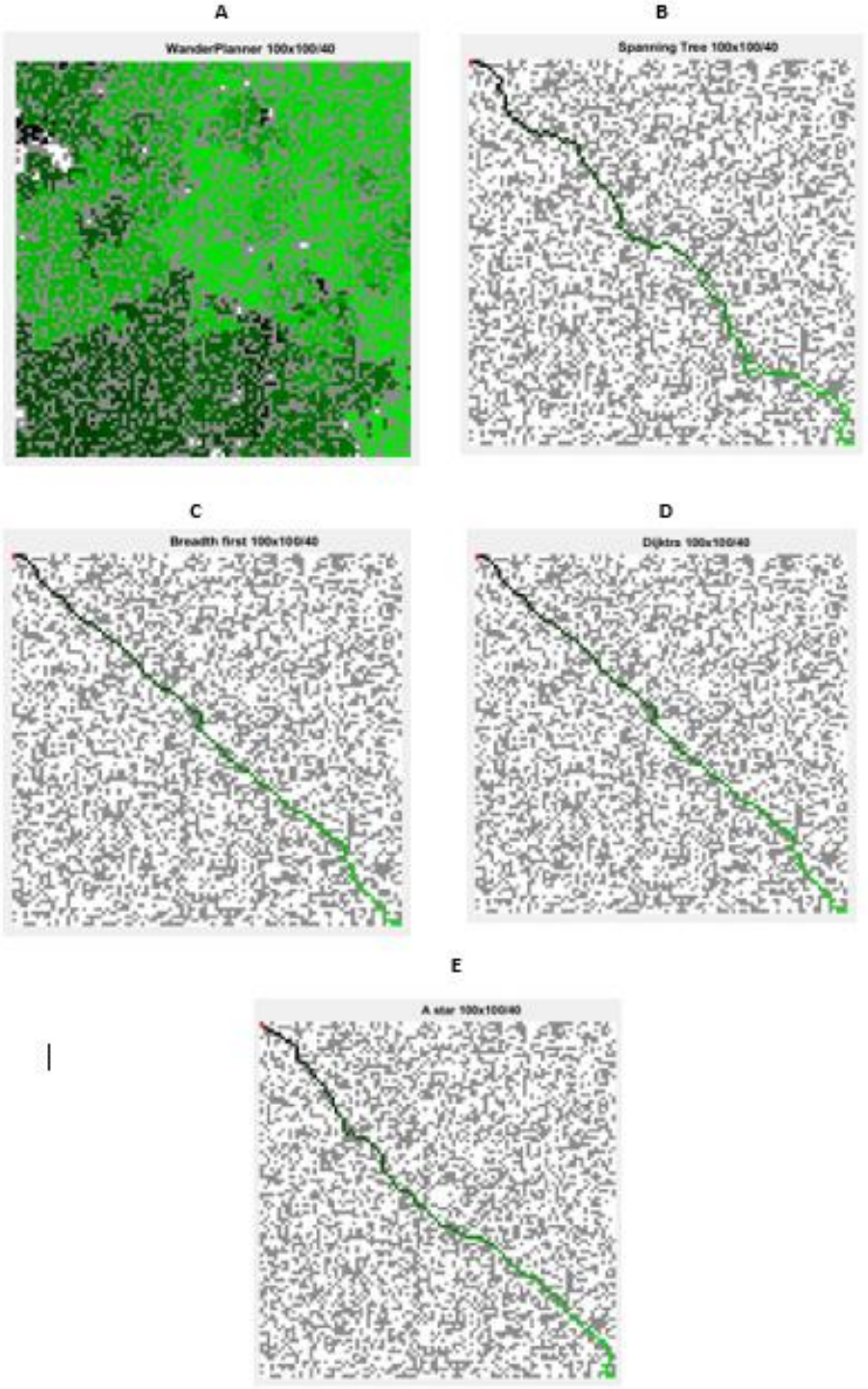
Tamaño de Grilla 100 por 100 con un 20 por Ciento de Obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

Figura 23

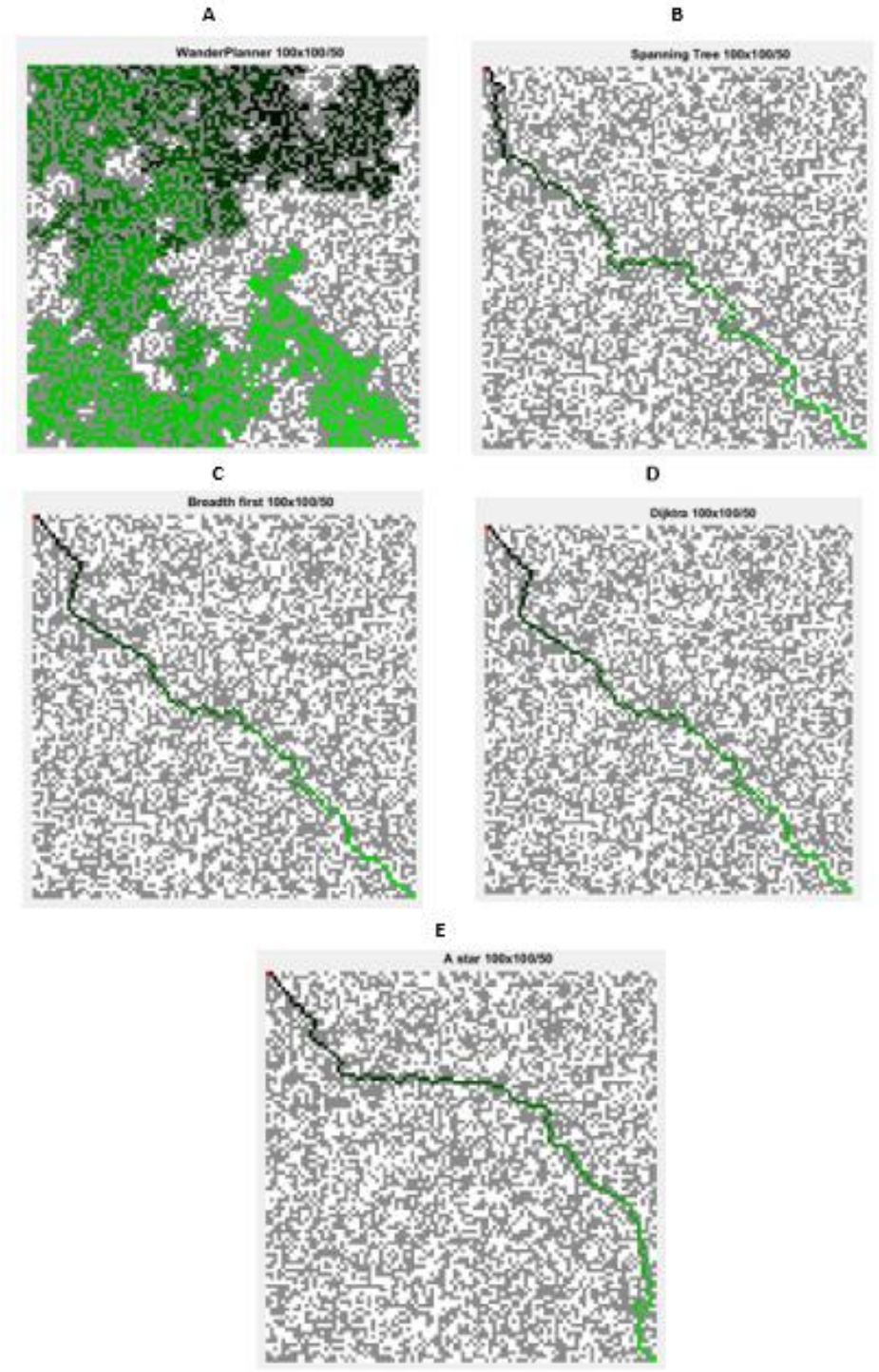
Tamaño de Grilla 100 por 100 con un 40 por Ciento de Obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

Figura 24

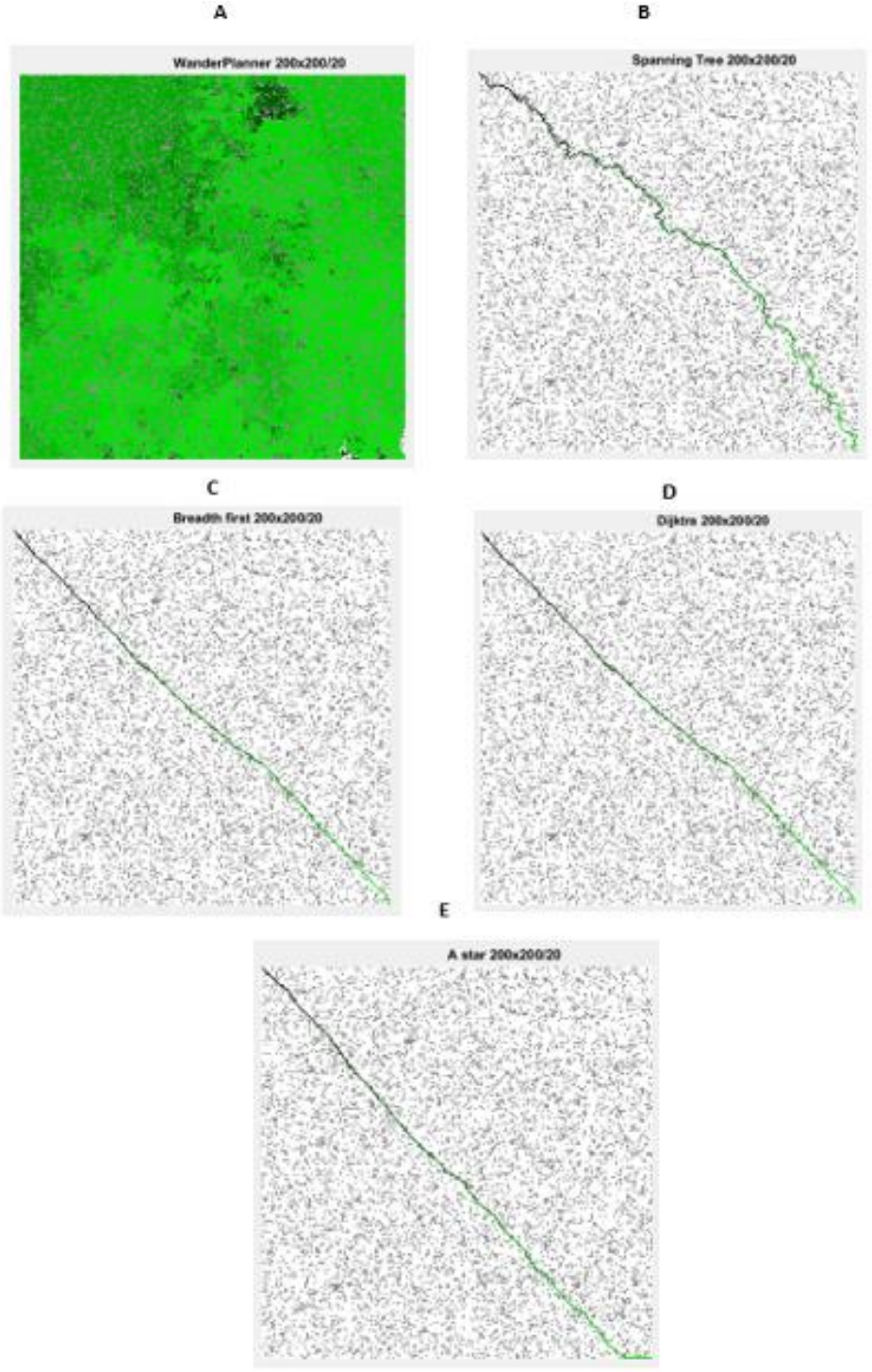
Tamaño de Grilla 100 por 100 con un 50 por Ciento de Obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

Figura 25

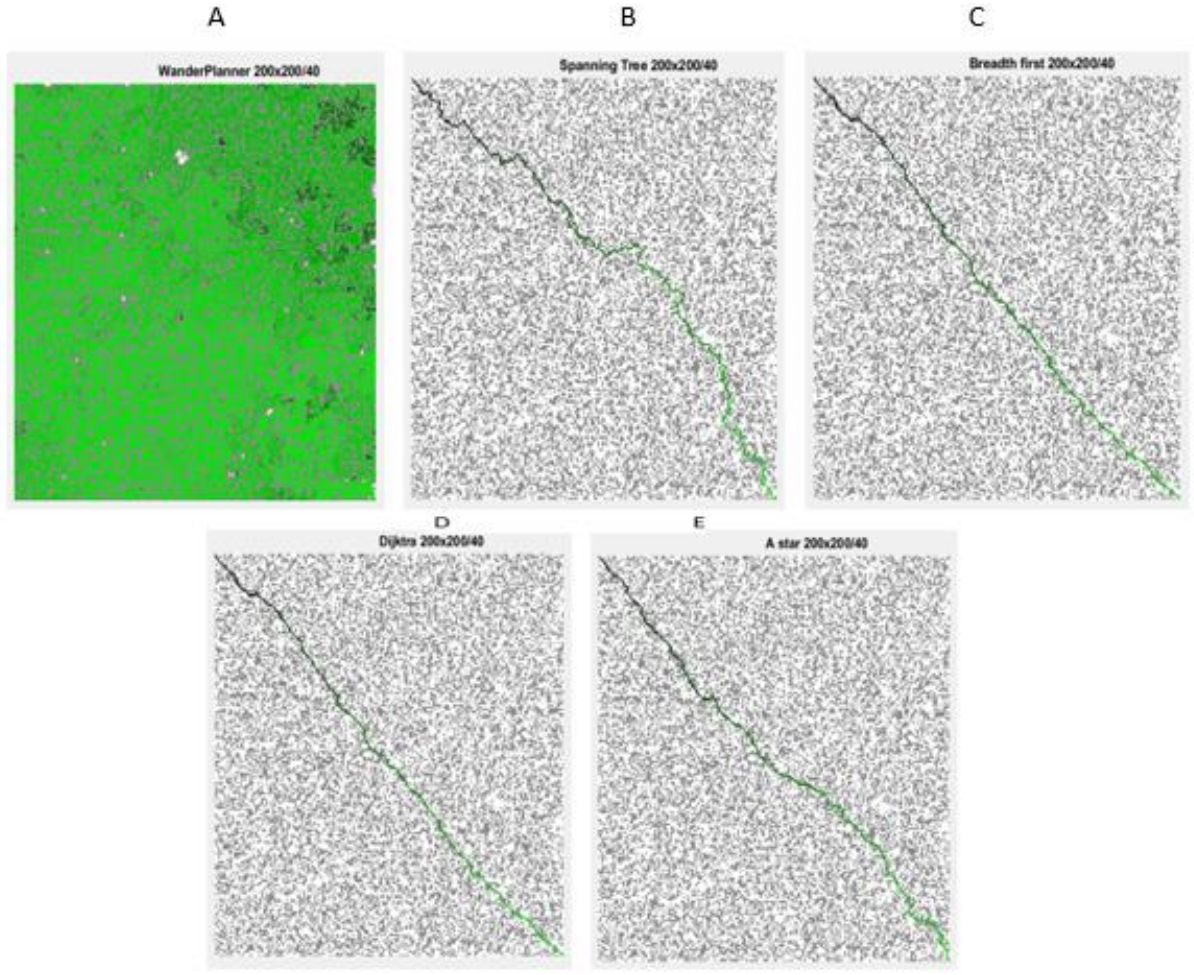
Tamaño de Grilla 200 por 200 con un 20 por Ciento de Obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

Figura 26

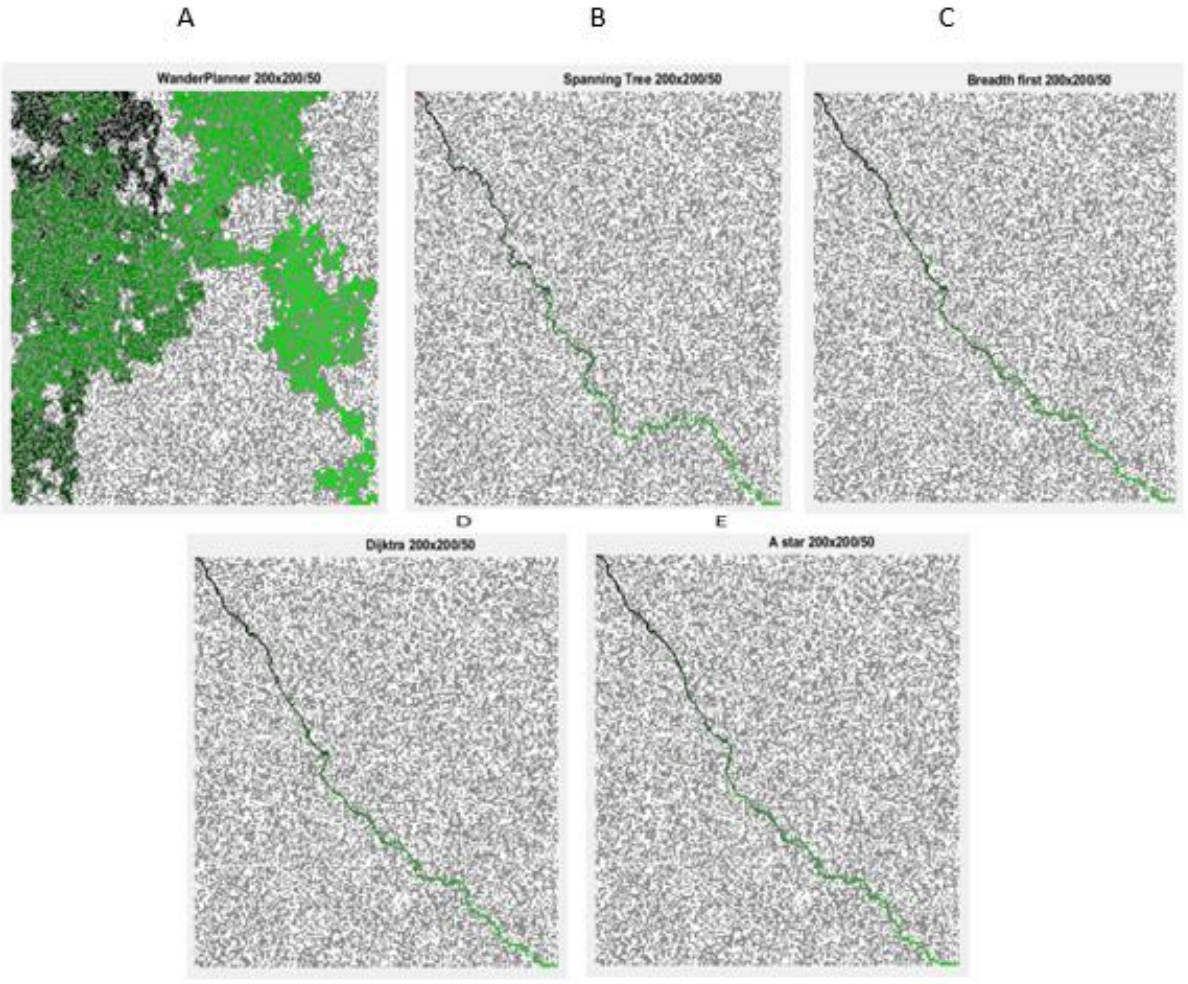
Tamaño de Grilla 200 por 200 con un 40 por Ciento de Obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

Figura 27

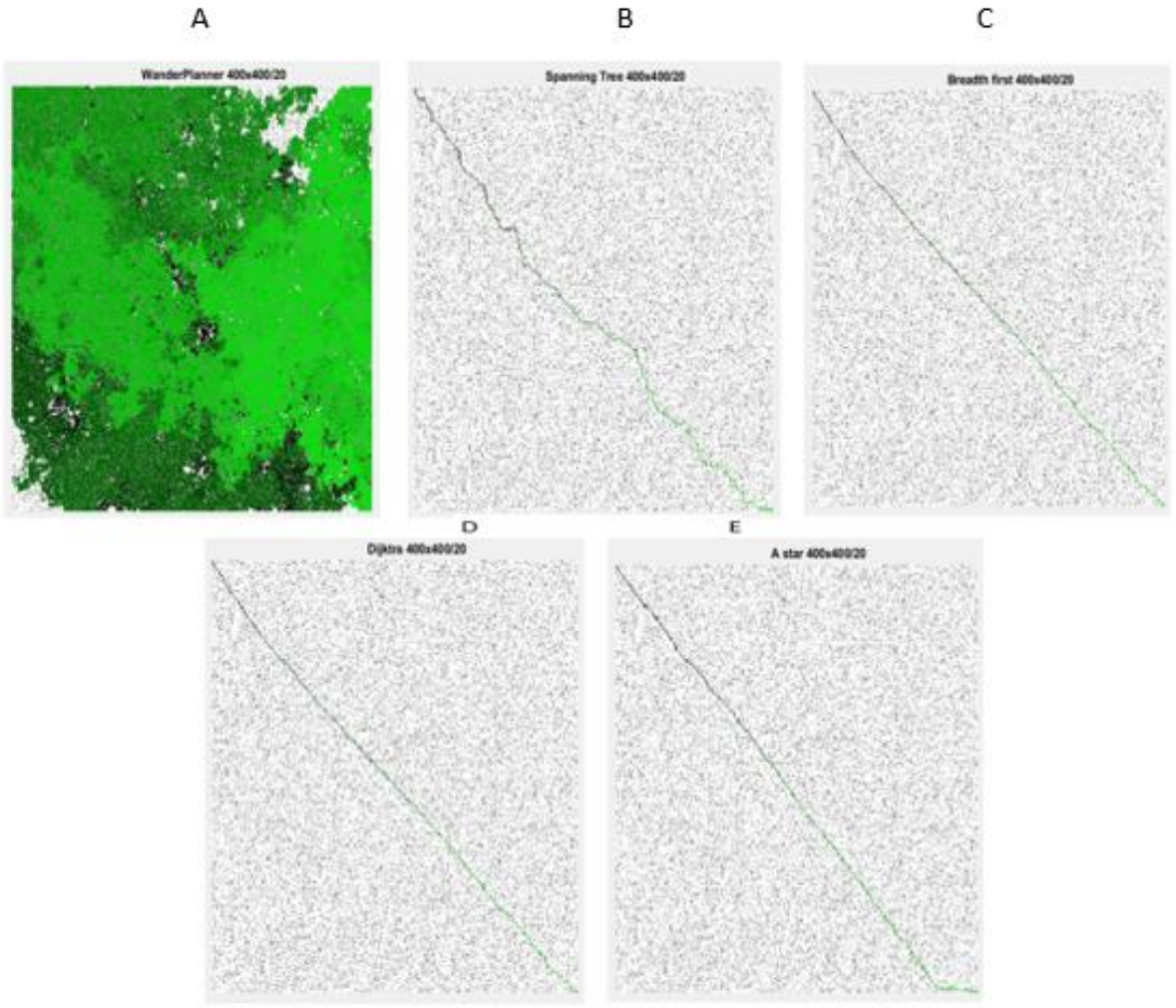
Tamaño de Grilla 200 por 200 con un 50 por Ciento de obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

Figura 28

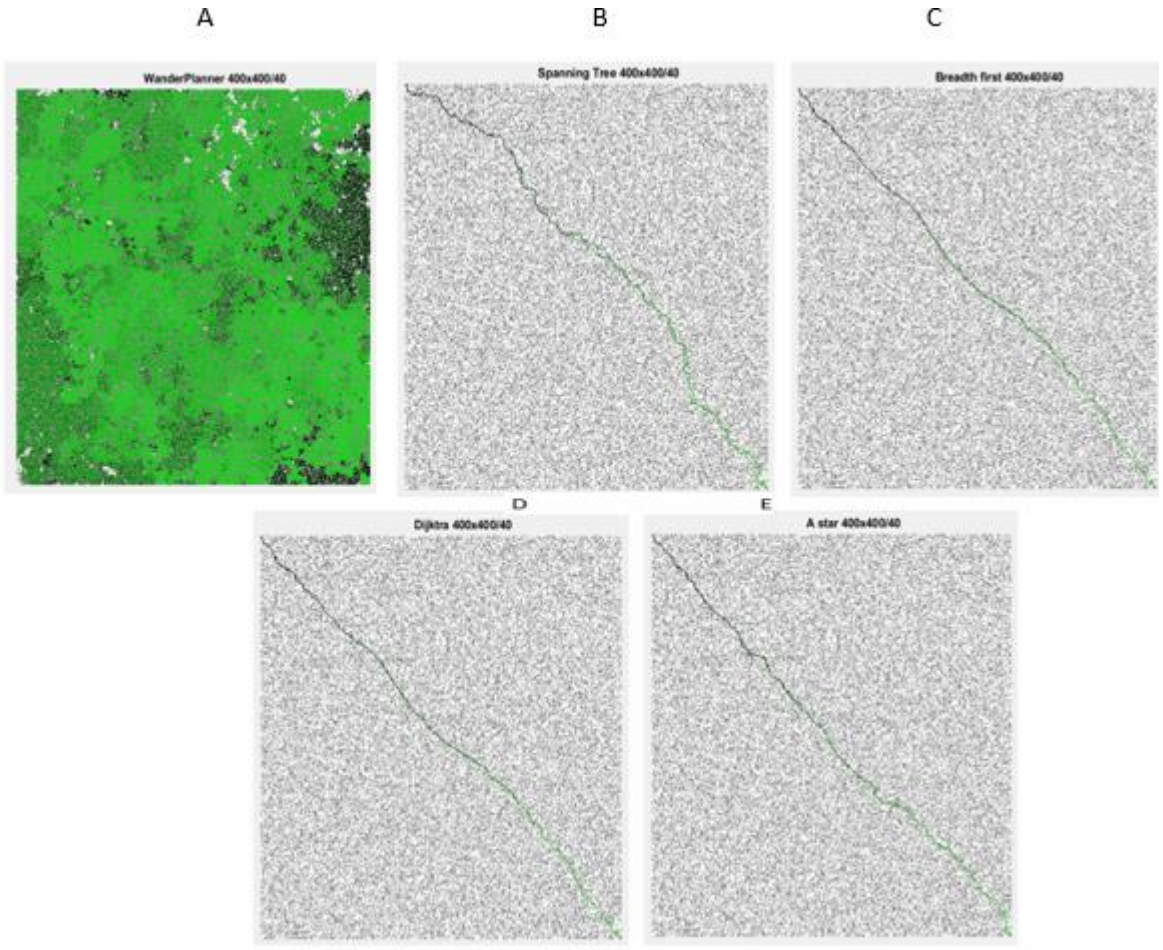
Tamaño de Grilla 400 por 400 con un 20 por Ciento de obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

Figura 29

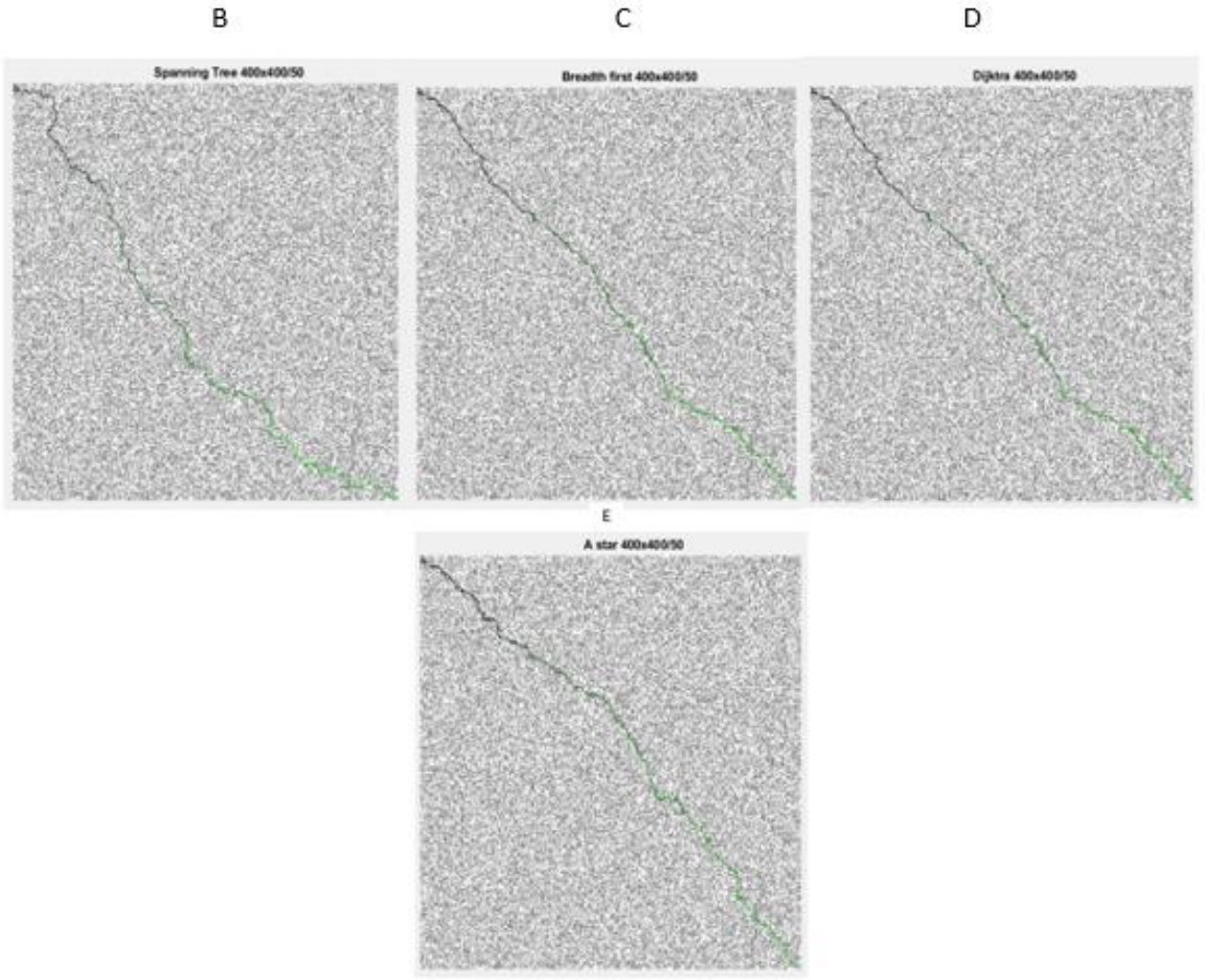
Tamaño de Grilla 400 por 400 con un 40 por Ciento de Obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

Figura 30

Tamaño de Grilla 400 por 400 con un 50 por Ciento de Obstáculos. Software Matlab



Fuente. Elaboración propia del documento (2023).

En la grilla 400 X 400 con 50 por ciento de obstáculos el primer algoritmo *wander planner* demoró más de 20 horas en su ejecución completa. Por esta razón en la Figura 30 no se presenta la sección A.

Comparación de Algoritmos Implementados

En la Tabla 7 se muestran los resultados obtenidos de cada algoritmo al evaluar los cinco parámetros de desempeño. Por cada parámetro se obtuvieron nueve datos los cuales ayudan a evaluar el comportamiento de cada algoritmo con respecto a los otros. Se debe resaltar que para obtener los datos correspondientes se realizaron las pruebas pertinentes en las mismas situaciones y condiciones del espacio de trabajo. Se muestra el desempeño de los algoritmos con los tres tamaños de grillas evaluados y los tres porcentajes de obstáculos con respecto a los criterios de evaluación y comparación: Tiempo total, número de iteraciones, número de celdas expandidas y número de celdas del camino. El criterio estado finalizado se generaliza, todos finalizan con éxito, porque en las pruebas realizadas se aseguró la llegada al objetivo.

Las notaciones utilizadas en la Tabla 7 de resultados son: WP *Wander planner*, ST Spanning Tree, BFS *Breadth First*, DIJ *Dijkstra y* A* *A start.*

Figura 31

Resultados: Tiempo total, número de iteraciones, número de celdas del camino y número de celdas del camino

Tamaño de la grilla	Porcentaje de obstáculos	Tiempo Total				
		WP	ST	BFS	DIJ	A*
100x100	20	207,9549	1,5287	2,0936	1,8033	0,2255
	40	219,0304	1,5242	2,1327	1,7694	0,0608
	50	219,5686	1,4648	1,9805	1,7015	0,4416
200x200	20	4.959,0558	36,1513	45,3923	78,0303	0,6358
	40	18.926,5097	27,7975	47,0005	89,5253	0,7533
	50	18.951,4939	28,4269	46,7572	92,6404	1,0010
400x400	20	13.088,9730	602,3972	1.498,3725	1.431,6956	2,8389
	40	58.437,0409	1.278,7884	1.141,6798	1.229,0078	2,9418
	50		637,2747	1.703,9499	1.209,2056	5,7154
		Número de iteraciones				
		WP	ST	BFS	DIJ	A*
100x100	20	263276	7992	7998	7999	132
	40	263276	7992	7998	7999	187
	50	263276	7992	7998	7999	1631
200x200	20	1161812	31965	31996	31999	1631
	40	2206339	31965	31996	31999	1631
	50	2206339	31965	31996	31999	1631
400x400	20	1738074	127990	127997	127998	1631
	40	2805938	127990	127997	127998	1631
	50		127990	127997	127998	1980
		Número de celdas expandidas				
		WP	ST	BFS	DIJ	A*
100x100	20	263278	7998	7999	8000	404
	40	190833	5989	5990	5991	379
	50	54967	4902	4908	4910	1094
200x200	20	1161814	31999	31999	32000	834
	40	2206341	23939	23935	23940	803
	50	253383	19622	19621	19623	1088
400x400	20	1738076	127998	127998	127999	1756
	40	2805940	95839	95839	95845	1668
	50		78599	78644	78648	2221
		Número de celdas del camino				
		WP	ST	BFS	DIJ	A*
100x100	20	263278	147	107	107	117
	40	190833	142	118	118	128
	50	54967	159	137	137	160
200x200	20	1161814	303	215	215	224
	40	2206341	308	241	241	258
	50	253383	327	275	275	278
400x400	20	1738076	584	428	428	460
	40	2805940	620	486	486	522
	50		663	556	556	594

Fuente. Elaboración propia del documento (2023).

Comparación según cada Criterio con Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos

Para realizar esta comparación se tomaron los datos obtenidos en los ejercicios con el 40 % de los obstáculos, ya que el resultado con los otros porcentajes es semejante. Este análisis ayuda a concluir el comportamiento de los algoritmos según cambia el tamaño del espacio de trabajo y definir cuál de los cinco algoritmos para entornos estáticos estudiados es el mejor según la calificación de los parámetros de desempeño.

Se calcula una calificación por cada parámetro evaluado a cada algoritmo, dando una puntuación máxima al que tenga el mejor valor y una proporción equivalente a los demás algoritmos. Al final se obtiene un promedio que determina la eficiencia de cada uno donde el mayor valor será el del algoritmo que presente el mejor desempeño.

Se evidencia en la Tabla 7 que los resultados de *Wander planner* son muy grandes en comparación a los datos de los otros cuatro algoritmos, esto hace que en una escala lineal no se observe la diferencia entre los demás.

El primer parámetro estudiado es el tiempo, en la Tabla 8 se muestran los valores obtenidos en segundos para los 5 algoritmos, y en la Figura 31 se observan los resultados gráficamente.

Figura 32

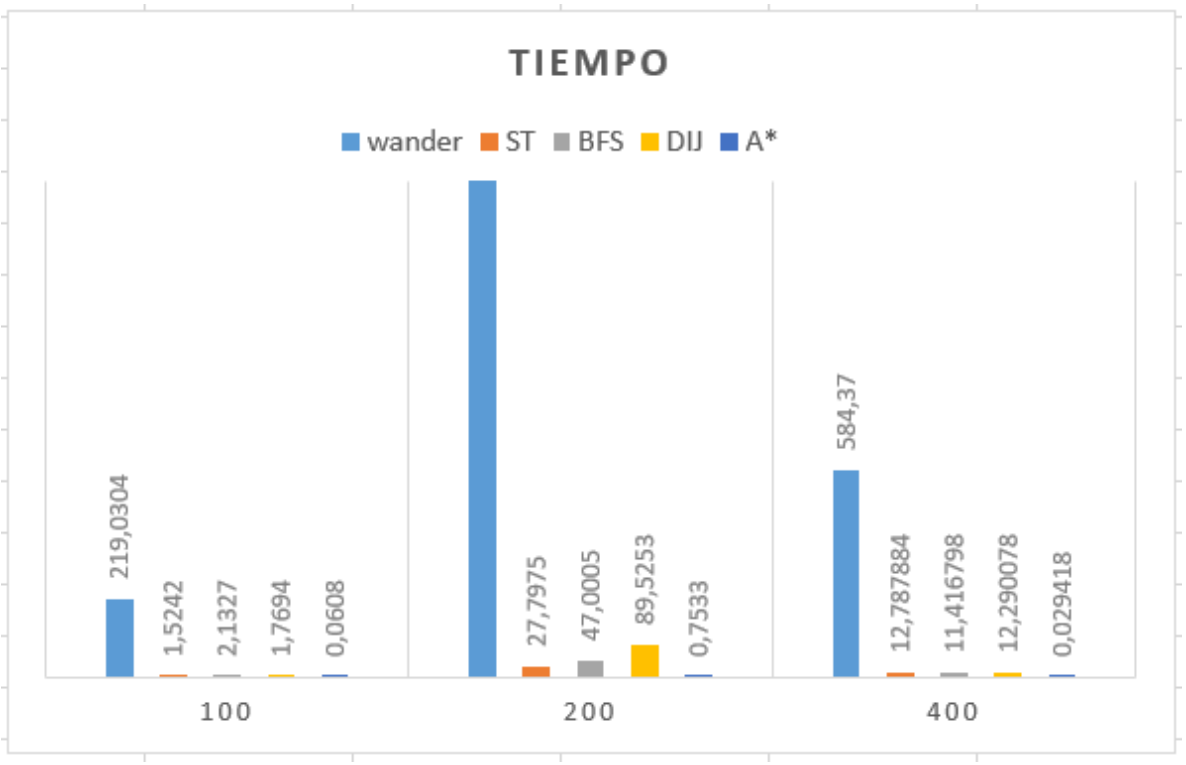
Tiempo. Tamaño de grilla 100, 200, 400 y 40% de Obstáculos

Tiempo					
	algoritmo				
	wander	ST	BFS	DIJ	A*
100	219,0304	1,5242	2,1327	1,7694	0,0608
200	18926,5097	27,7975	47,0005	89,5253	0,7533
400	58437	1278,7884	1141,6798	1229,0078	2,9418

Fuente. Elaboración propia del documento (2023).

Figura 33

Gráfica de Barras para el Tiempo. Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos



Fuente. Elaboración propia del documento (2023).

La gráfica de tiempo en el tamaño de grilla 400 se encuentra escalada a 100 para poder ver claramente el resultado. Se nota en el parámetro tiempo que el primer algoritmo *wander planner* muestra un valor muy elevado, se puede deducir que el costo computacional es muy alto. Los tres algoritmos siguientes ST, BFS y DIJ permanecen en una línea intermedia con algunas diferencias entre ellos. Pero la gran diferencia se la vuelve a notar en el algoritmo A* ya que sus tiempos son muy reducidos a comparación de los demás. Esto se debe a que su objetivo principal es realizar el proceso con el menor gasto computacional posible.

El segundo parámetro estudiado es el número de iteraciones. Los datos conseguidos se encuentran en la Tabla 9 para los 5 algoritmos, y luego en la Figura 32 se observan los resultados gráficamente.

Figura 34

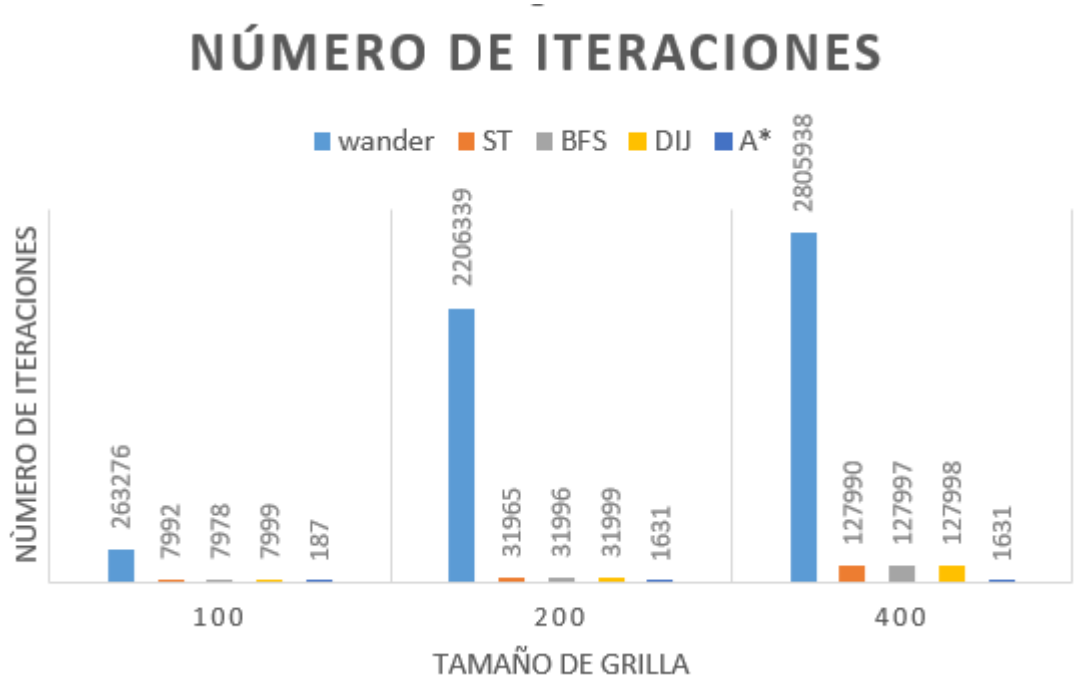
Número de Iteraciones. Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos

Número de iteraciones					
	algoritmo				
	wander	ST	BFS	DIJ	A*
100	263276	7992	7978	7999	187
200	2206339	31965	31996	31999	1631
400	2805938	127990	127997	127998	1631

Fuente. Elaboración propia del documento (2023).

Figura 35

Gráfica de Barras. Número de Iteraciones. Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos



Fuente. Elaboración propia del documento (2023).

De acuerdo con el parámetro número de iteraciones, se sigue observando la ineficiencia del algoritmo *wander planner*, el valor arrojado es demasiado elevado, pero es este uno de los parámetros que muestra que A* star es más eficiente ya que mientras menor sea el número de iteraciones menor será el costo computacional.

Cómo tercer parámetro estudiado se encuentra el número de celdas expandidas. En la Tabla 10 se puede observar los datos conseguidos para los 5 algoritmos, y luego en la Figura 33 se indican los resultados gráficamente.

Figura 36

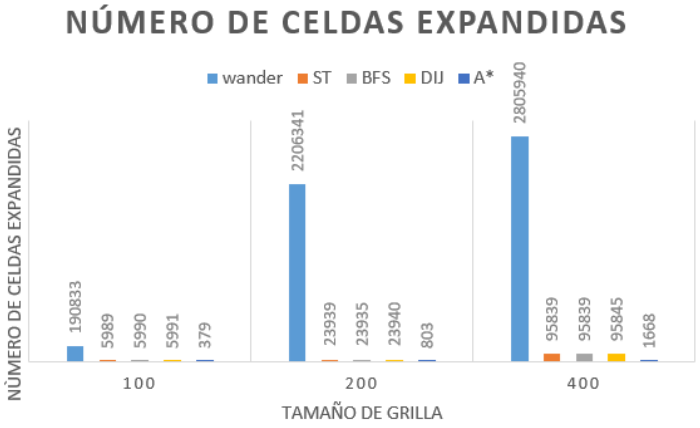
Celdas Expandidas. Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos

Número de celdas expandidas					
	algoritmo				
	wander	ST	BFS	DIJ	A*
100	190833	5989	5990	5991	379
200	2206341	23939	23935	23940	803
400	2805940	95839	95839	95845	1668

Fuente. Elaboración propia del documento (2023).

Figura 37

Gráfica de Barras. Celdas Expandidas. Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos



Fuente. Elaboración propia del documento (2023).

El parámetro número de celdas expandidas, muestra cuál de los algoritmos realiza menos análisis a las casillas y se vuelve a notar el mismo comportamiento de los anteriores resultados siendo *wander planner* el algoritmo que más casillas utiliza y se sigue afirmando la ineficiencia de éste. Después se encuentran los algoritmos ST, BFS y DIJ, que han mostrado resultados aproximados y al final se encuentra A* que vuelve a demostrar que está diseñado para reducir el computacional.

Ahora se analiza el parámetro número de celdas del camino. La Tabla 11 muestra los datos conseguidos para los 5 algoritmos, y luego en la Figura 34 se indican los resultados gráficamente.

Figura 38

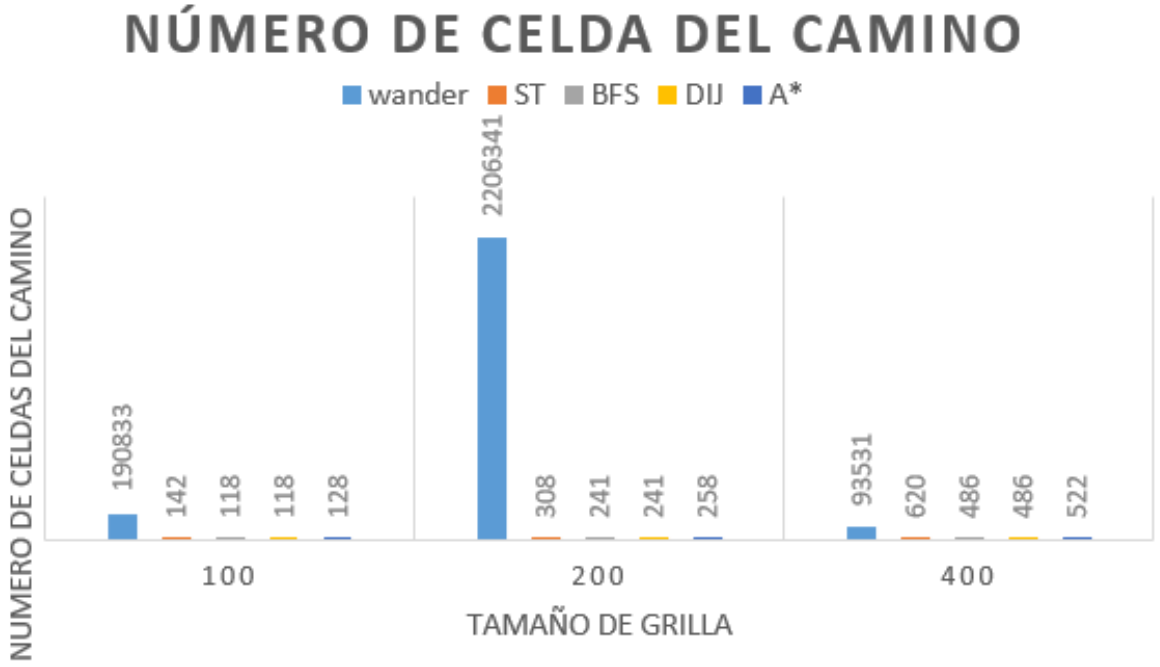
Cantidad de Celdas que Tiene la Trayectoria. Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos

Número de celda del camino					
	algoritmo				
	wander	ST	BFS	DIJ	A*
100	190833	142	118	118	128
200	2206341	308	241	241	258
400	2805940	620	486	486	522

Fuente. Elaboración propia del documento (2023).

Figura 39

Gráfica de Barras. Cantidad de Celdas que Tiene la Trayectoria. Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos



Fuente. Elaboración propia del documento (2023).

La ineficiencia de *Wander planner* es indiscutible de acuerdo con los resultados en el parámetro número de celdas del camino. Se aprecia claramente la similitud en los resultados de los últimos 4 algoritmos ST, BFS, DIJ y A*, pero es claro que BFS y DIJ presentan la ruta más corta.

Por último, el parámetro efectividad al llegar al objetivo. Las pruebas se realizaron en entornos donde siempre existió paso entre el punto inicial y el punto final.

Se efectuaron muchos intentos con los dos primeros algoritmos, dando como resultado que *Wander planner* tiene una efectividad baja para alcanzar el objetivo por lo que no se tiene en cuenta para la evaluación.

Los últimos 4 algoritmos ST, BFS, DIJ y A*, presentan una eficiencia del cien por ciento lo cual nos garantiza que, sin importar el espacio de trabajo, mientras exista camino entre los dos puntos, estos lograrán encontrar la trayectoria. Además, son capaces de definir o informar si no existe paso, esto los hace muy eficientes ya que proporcionan toda la información necesaria para resolver el problema.

Evaluación según cada Criterio con Tamaño de Grilla 100, 200, 400 y 40% de Obstáculos

De acuerdo con el proceso realizado se concluye que la eficiencia de un algoritmo depende en conjunto de los datos obtenidos en cada parámetro de desempeño, ya que no se puede decretar con un solo parámetro cuál es el mejor. Se analizó la manera de sacar una puntuación total y que esté basada en las puntuaciones parciales, y con esos resultados decidir cuál de los algoritmos tiene el mejor desempeño.

Se propone que la calificación se obtendrá referenciando inicialmente el mejor algoritmo en cierto parámetro con un valor de 10, luego utilizando la fórmula de la regla de tres simples inversas (ya que se trata de magnitudes inversamente proporcionales) se obtiene la puntuación de los algoritmos según el resultado en cada criterio de evaluación. En la Tabla 12 se presentan las calificaciones parciales de cada algoritmo, también los promedios por algoritmo en cada tamaño de grilla y por último la puntuación total.

Figura 40

Cuadro de Calificaciones de acuerdo con los 4 Parámetros de Desempeño. Tamaño de Grilla 100, 200, 400 Y 40% de Obstáculos

ALGORITMO	TAMAÑO DE GRILLA	TIEMPO	# ITERACIONES	CELDAS EXPANDIDAS	CELDAS DEL CAMINO	PROMEDIO	PUNTUACIÓN TOTAL
A*	100	10	10	10	9,2188	9,8047	**9,8225**
	200	10	10	10	9,3411	9,8353	
	400	10	10	10	9,3103	9,8276	
DIJ	100	0,3436	0,2338	0,6326	10	2,8025	**2,7054**
	200	0,0841	0,5097	0,3354	10	2,7323	
	400	0,0239	0,1274	0,1740	10	2,5813	
BFS	100	0,2851	0,2344	0,6327	10	2,7881	**2,7071**
	200	0,1603	0,5098	0,3355	10	2,7514	
	400	0,0258	0,1274	0,1740	10	2,5818	
ST	100	0,3989	0,2340	0,6328	8,3099	2,3939	**2,2233**
	200	0,2710	0,5102	0,3354	7,8247	2,2353	
	400	0,0230	0,1274	0,1740	7,8387	2,0408	
WANDER	100	0,0028	0,0071	0,0199	0,0062	0,0090	**0,0052**
	200	0,0004	0,0074	0,0036	0,0011	0,0031	
	400	0,0005	0,0058	0,0059	0,0017	0,0035	

Fuente. Elaboración propia del documento (2023).

Según las calificaciones mostradas en la Tabla 12 se concluye que el mejor algoritmo para entornos estáticos es A*, notando que en la mayor cantidad de parámetros su calificación es muy alta y a pesar de que al observar las celdas del camino los algoritmos DIJ y BFS muestran un mejor resultado, la diferencia entre los tres no es muy grande o significativa para influir en la puntuación final.

Pruebas Iniciales Tomando Grillas de Ocupación en un Punto de Inicio y Otro Destino. Entornos Dinámicos

En el caso de los entornos dinámicos se realizó la comparación del algoritmo D star y A star From Scratch los cuales calculan la ruta y siguen el recorrido para hacer la replanificación en los casos que sean necesarios. Para estas pruebas se tomó una grilla cuadrada de cien celdas por lado y un 45% de obstáculos en el espacio de trabajo. El punto inicial se ubicó en la esquina superior izquierda y el punto final se encuentra en la esquina inferior derecha para exigir al

algoritmo lo máximo posible. Se tomó la decisión de realizar dos cambios en el espacio de trabajo y con ello poder comprobar el *replanning* y observar el comportamiento de los dos algoritmos a estos cambios.

Es muy importante aclarar que la función de estos algoritmos diseñados para entornos dinámicos no es solo calcular la ruta, sino que deben acompañar el recorrido del usuario u objeto a la meta y mientras esto sucede, analizan si existe cambio en el entorno y cuando detecten alguna diferencia recalculan la ruta para asegurar que éste llegará al punto de destino. Esto con el fin de poderse adaptar a situaciones reales.

En las Figuras 35 y 36 de las trayectorias se puede observar tres secciones A, B y C donde la sección A hace referencia a la ruta calculada inicialmente, la sección B muestra la trayectoria después que el espacio de trabajo tuviera el primer cambio y por último en la sección C se encuentra la trayectoria luego de la segunda replanificación. La trayectoria calculada tiene un color verde y las casillas que fueron recorridas tendrán un color azul, esto ayuda a identificar cómo va el recorrido del usuario hasta que se encuentra un cambio.

Al observar la Figura 35 obtenida en el algoritmo D star se nota muy bien cómo realiza la replanificación a partir del punto donde se detectó el cambio, pero mantiene toda la información anterior y con esto se logra conseguir un análisis completo de cómo se realizó el proceso desde su inicio.

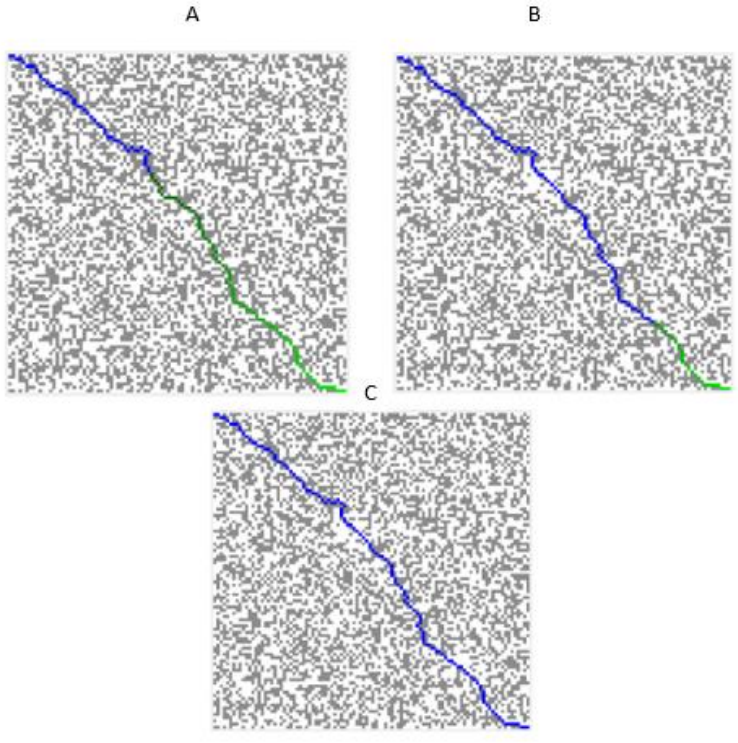
Por el contrario, la Figura 36 sección A obtenida por el algoritmo *A star From Scratch* muestra claramente cómo calcula el recorrido y al encontrar un cambio en el espacio de trabajo toma el último punto donde quedó como su nuevo punto inicial y a partir de este calcula la ruta hasta el destino sin guardar la información anterior, por lo que la Figura 36 sección C solo

muestra un trayecto muy corto y no se puede observar cómo fue el proceso y tampoco se logra obtener una información clara de cómo llegó al objetivo.

Fue necesario hacer muchas pruebas para obtener un espacio de trabajo donde se garantiza que existe un camino hasta el punto final en todo momento, refiriéndose a que si existe algún cambio en el proceso también llegue al punto destino. Esto se debe a que *A star from scratch* tiene sus bases en un algoritmo para entornos estáticos y si no encuentra ruta no puede continuar. En cambio, D star está diseñado para superar cualquier situación, aun cuando no hay paso, porque es un algoritmo muy robusto y muy bien diseñado. Al hacer las pruebas se pudo notar que cuando se aumenta el porcentaje de obstáculos *A star from scratch* tiende a fallar con mucha más frecuencia mientras que D star siempre encontrará la solución y llegará al objetivo.

Figura 41

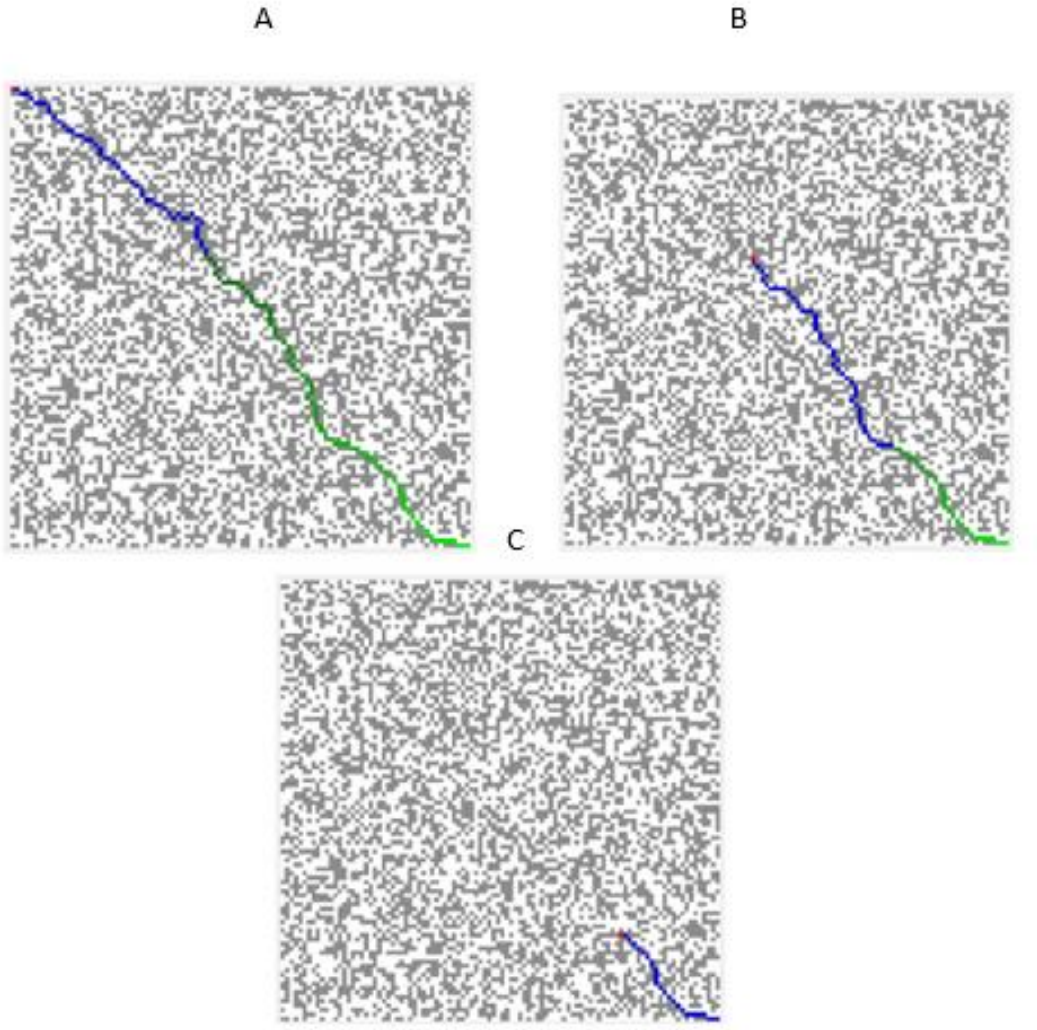
Algoritmo D. Tamaño de Grilla 100 por 100 con un 45 por Ciento de Obstáculos*



Fuente. Elaboración propia del documento (2023).

Figura 42

Algoritmo A star from Scratch. Tamaño de Grilla 100 por 100 con un 45 por Ciento de Obstáculos



Fuente. Elaboración propia del documento (2023).

Comparación de Algoritmos Implementados

Para realizar la comparación entre los dos algoritmos se tienen en cuenta los cinco parámetros de desempeño con los cuales se podrá dar un concepto de su comportamiento a lo largo de las pruebas.

En la Tabla 13 se exponen los resultados de los datos por cada proceso para calcular el camino, teniendo en cuenta los dos *replanning*, y en las figuras 37 y 38 se podrá ver una comparación de los resultados gráficamente. Los datos totales se consiguen al realizar la suma de las cifras parciales de cada parámetro, lo que nos permite hacer un paralelo del comportamiento de cada algoritmo.

Figura 43

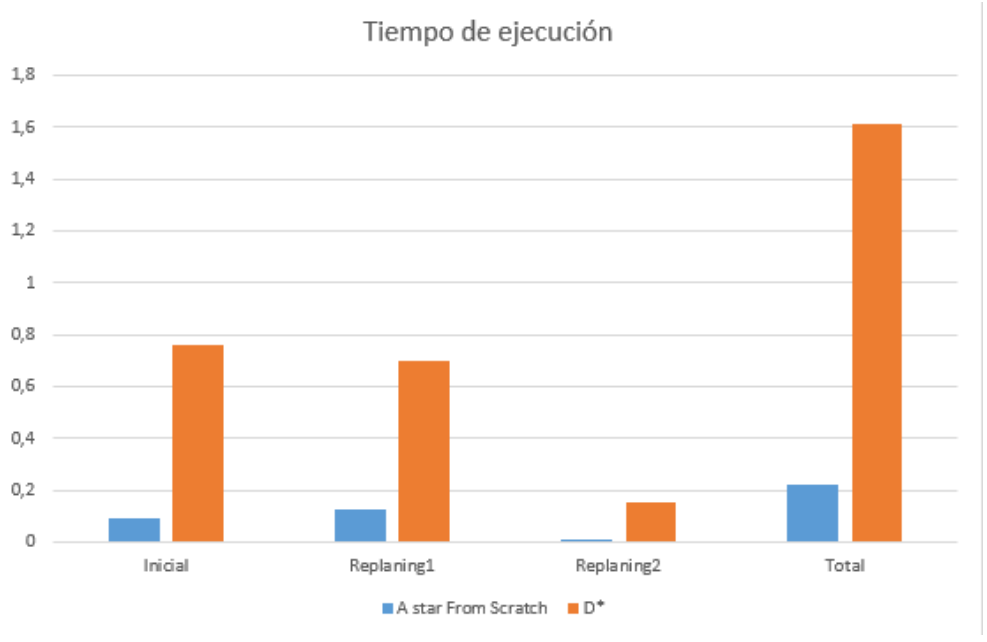
Resultados Parámetros Parciales y Totales.

Algoritmo	Parámetro	Inicial	Replaning1	Replaning2	Total
A star From Scratch	Tiempo	0,0925	0,1225	0,0065	**0,2215**
	Celdas del Camino	50	54	26	**130**
	Celdas Expandidas	400	235	81	**716**
D*	Tiempo	0,761	0,6996	0,1532	**1,6138**
	Celdas del Camino	50	54	26	**130**
	Celdas Expandidas	684	14	1	**699**

Fuente. Elaboración propia del documento (2023).

Figura 44

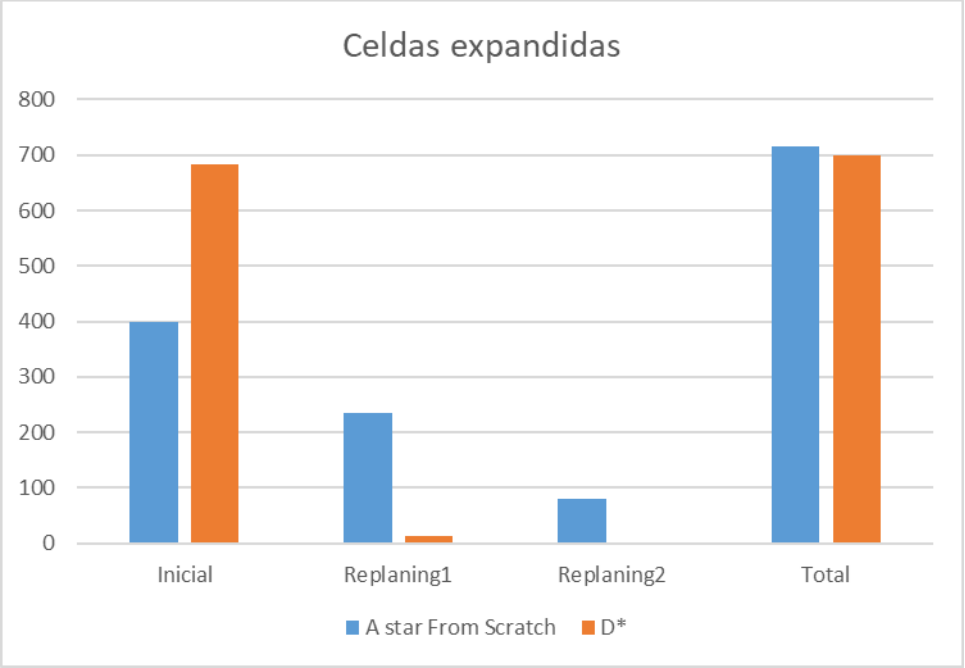
Tiempo de Ejecución



Fuente. Elaboración propia del documento (2023).

Figura 45

Celdas Expandidas.



Fuente. Elaboración propia del documento (2023).

De acuerdo con la figura 37, donde se muestran los tiempos parciales para D star, es claro que decrecen, ya que este aprovecha los datos obtenidos y avanza modificando únicamente las casillas que lo necesiten, disminuyendo el tiempo en el cálculo de datos. *A star From Scratch* realiza tres ejercicios desde cero, lo que se refiere a que el tiempo 1 representa una trayectoria, el tiempo 2 una trayectoria diferente y el tiempo 3 otra, estos dependen de las condiciones del espacio de trabajo. El tiempo total de D* es mayor porque su estructura tiene una composición más compleja y es diseñada para enfrentar cualquier condición en el espacio de trabajo y terminar con éxito. El tiempo total de *A star From Scratch* es menor por ser más básico y estar basado en A* que es diseñado para entornos estáticos.

En la Figura 38 encontramos que D* expande menos celdas porque aprovecha los datos guardados, inicialmente se obtiene una cantidad muy grande de casillas pero en los siguientes

momentos se observa un decrecimiento exponencial en la cantidad expandida. Por el contrario, el algoritmo A star From Scratch al momento de hacer un *replanning* vuelve a evaluar las casillas necesarias y puede ser que anteriormente ya se hayan expandido lo que aumentará el valor de este parámetro y muestra un decrecimiento lineal.

Número de Celdas del Camino.

Este parámetro da resultados iguales en los dos algoritmos implementados en entornos dinámicos. La razón es que los dos encuentran el mismo camino, lo que se observa claramente al comparar las figuras 35 y 36.

Número de Iteraciones.

Si se analiza la evolución en la implementación en tiempo real, se puede concluir que *A star From Scratch* realiza los procesos internos muchas veces, pero al ser un algoritmo basado en entornos estáticos la complejidad es menor. D star es un algoritmo mucho más robusto y los procesos internos requieren de mucho más trabajo computacional, lo que lleva a afirmar que este parámetro es incomparable ya que los dos no se encuentran en las mismas condiciones en su programación.

Efectividad al Llegar al Objetivo

Esta medida es la más importante, porque muestra verdaderamente cuál de los algoritmos logra su cometido.

Para determinar los datos se realizaron 20 pruebas. En *A star From Scratch* la efectividad sólo se obtuvo en una, en cambio D* llegó al objetivo en todas las pruebas.

D star tiene una efectividad del cien por ciento lo que se traduce en que consigue siempre su objetivo sin importar las condiciones del espacio de trabajo.

Para *A star From Scratch*, se debe garantizar que el espacio de trabajo en todo momento le permita encontrar una solución y esto no se puede asegurar nunca. Al aumentar la complejidad del espacio de trabajo o la cantidad de cambios en el recorrido, se disminuye la probabilidad de llevar al éxito el proceso. La efectividad es mínima.

Evaluación

De la misma manera que con entornos estáticos, se analizó la forma de obtener una puntuación total y que esté basada en las puntuaciones parciales, ya que la eficiencia de un algoritmo depende en conjunto de los datos obtenidos en cada parámetro. Con esos resultados se pudo definir cuál de los algoritmos tiene el mejor desempeño.

La calificación se consigue referenciando inicialmente el mejor algoritmo en este parámetro con un valor de 10, luego utilizando la fórmula de la regla de tres simples inversas, porque las magnitudes son inversamente proporcionales, se obtiene la puntuación de los algoritmos según el resultado en cada criterio de evaluación. En la Tabla 14 se presentan las calificaciones parciales de cada algoritmo y la puntuación total que se trata del promedio.

Figura 46

Cuadro de Calificaciones de acuerdo con los 4 Parámetros de Desempeño

Algoritmo	TIEMPO	EFECTIVIDAD	CELDAS EXPANDIDAS	CELDAS DEL CAMINO	PROMEDIO
A star From Scratch	10	0,5	9,7626	10	**7,5656**
D*	1,3725	10	10	10	**7,8431**

Fuente. Elaboración propia del documento (2023).

De acuerdo con la Tabla 14 se concluye que el algoritmo D* es superior en el promedio. En el análisis realizado la diferencia está ligada a la efectividad, ya que este *A star From Scratch* presenta una efectividad pequeña y variable, mientras que la efectividad de D* es alta y constante.

En el método se sugiere dar jerarquía a los criterios de desempeño para algoritmos en tiempo real y poder dar una puntuación total más acertada, según las necesidades de usuario y así determinar el mejor.

Del 100 por ciento de una calificación total se determina que la efectividad tenga un 50 % del total, porque la misión inicialmente es llegar al objetivo siempre. En la Tabla 15 se propone un cuadro de calificaciones con jerarquía donde el tiempo tendrá un porcentaje de 10, la efectividad tiene un porcentaje de 50, las celdas expandidas y las celdas del camino de un porcentaje de 20 cada una, para un total del 100 por ciento.

Figura 47

Cuadro de Calificaciones con Jerarquía de acuerdo con los 4 Parámetros de Desempeño

Algoritmo	TIEMPO 10%	EFECTIVIDAD 50%	CELDAS EXPANDIDAS 20%	CELDAS DEL CAMINO 20%	TOTAL
A star From Scratch	10	0,5	9,7626	10	**5,2025**
D*	1,3725	10	10	10	**9,1373**

Fuente. Elaboración propia del documento (2023).

Determinar el Algoritmo de Mejor Desempeño

Comparando los cinco algoritmos implementados que realizan el proceso en entornos estáticos se puede concluir que el más eficiente es A star, a pesar de que es un algoritmo que posee más complejidad en comparación con los otros, porque no solo da una respuesta clara a las situaciones presentadas, sino que sus trayectorias son óptimas y también el costo computacional es mucho menor.

Cuando se trata de entornos dinámicos el algoritmo D star es el que tiene mejor desempeño, considerando los cinco criterios propuestos. Este algoritmo posee una mayor complejidad en su proceso, pero con esto garantiza que siempre encuentra un camino óptimo sin importar las situaciones que encuentre en su recorrido, aun en el caso de encontrarse encerrado

siempre espera una replanificación que asegura llegar al final, no importa el número de grillas, ni tampoco el porcentaje de obstáculos, funciona para todos los casos.

Conclusiones con los Resultados Obtenidos

Los algoritmos que realizan un proceso previo de análisis del espacio de trabajo antes de escoger la ruta, como *breadth first, Dijktra* y A star, resultan ser mucho más eficientes que los algoritmos que escogen la ruta a medida que expanden los nodos. Esto se debe a que con el proceso de análisis se pueden determinar diferentes factores como que no exista paso para la trayectoria mientras que *Wanderplanner* o *Spanning tree* no tienen la capacidad de determinar si hay paso o no hacia la meta antes de que finalicen el proceso.

En el proceso para determinar la trayectoria, teniendo en cuenta los datos de la lista *Close*, se puede concluir que es mejor buscar el camino desde el punto final y realizando búsqueda de los padres y no lo contrario ya que un nodo hijo puede tener un solo nodo padre pero un nodo padre puede tener más de un nodo hijo, eso garantiza encontrar el camino óptimo de una manera más simple.

El algoritmo *Wander planner* presenta un problema en el caso de no haber paso para llegar desde el punto de partida hasta el nodo final. Este algoritmo se desbordaría ya que da muchas vueltas o es cíclico y solo termina cuando encuentra el fin, en otras palabras, en el caso presentado, nunca finalizaría y quedaría repitiendo las celdas que tenga disponibles.

Cuando se trabaja en terrenos planos se puede observar que las trayectorias entre los tres últimos algoritmos (*Breadth first, Dijktra y A star*) son muy semejantes. Las mayores diferencias entre ellos están en el tiempo y el número de celdas expandidas.

De veinte pruebas realizadas el algoritmo D* obtuvo el 100 por ciento de efectividad calculando la trayectoria, mientras que el algoritmo de *A star From Scratch* obtuvo únicamente

un cero punto cinco de efectividad, dado que, si no existe trayectoria segura hacia el objetivo, el algoritmo no tiene posibilidad de resolver o encontrar la solución.

Conclusiones

De acuerdo con el análisis diagnóstico de la problemática de los invidentes en Colombia y en el mundo se puede concluir que la población es muy amplia y necesita ayuda. Los estudios hechos para esta población son muchos y la ayuda a su movilidad es una prioridad. En la actualidad, la cantidad de asistentes creados para mitigar esta problemática es muy amplia y su adquisición depende de los medios económicos.

El estudio nos lleva a asegurar que los algoritmos de planificación de trayectorias se han convertido en una herramienta muy importante para desarrollar dispositivos para la asistencia en movilidad.

En relación con los algoritmos estudiados se concluye que fueron los adecuados teniendo en cuenta que el nivel de dificultad iba aumentando y así se veía claramente cuál era el mejor. Los resultados obtenidos nos aseguran que los criterios de evaluación, el número de algoritmos implementados y el lenguaje de programación fueron una buena elección.

Con respecto a la implementación de los algoritmos, aunque fue un trabajo largo y de mucho esfuerzo se logra llegar a la conclusión exacta de cuáles son los algoritmos más eficientes en entornos dinámicos y estáticos. El resultado concreto que se obtuvo es que de los 5 algoritmos estudiados en entornos dinámicos el mejor es A* teniendo una calificación total de 9.87. Comparando el desempeño se observa claramente que la diferencia es muy clara en las casillas expandidas, ya que A* expande mucho menos y logra llegar al objetivo con mucha más efectividad. Con respecto a los dos algoritmos implementados en entornos dinámicos se concluye que D* es un algoritmo muy completo que asegura desde cualquier situación la efectividad al llegar al objetivo, su desempeño es calificado desde un conjunto de parámetros, teniendo la puntuación total de 9.1373.

Del método propuesto se concluye claramente que representa una gran ayuda para cualquier investigador que desee identificar los pasos que se deben seguir para concluir qué algoritmos de planificación de trayectorias son los más eficientes en cualquier ámbito de apoyo a la movilidad.

Recomendaciones

Este estudio da pie a nuevas investigaciones para asistir a las personas con discapacidades motrices, inconvenientes con sus manos, brazos y piernas, también con enfermedades como Alzheimer, deterioro cognitivo leve o demencia, siendo la base para estudios posteriores que apoyen con dispositivos y que utilicen la navegación intencionada, asegurando una buena movilidad y logrando una mejor calidad de vida.

Se recomienda tener en cuenta los pasos sugeridos en el método para asegurar una elección acertada de algoritmos de planificación de trayectorias.

Para trabajos futuros se recomienda la implementación de más algoritmos para entornos dinámicos, puesto que en este trabajo de grado se evaluaron únicamente dos en tiempo real y las condiciones actuales frente a las nuevas tecnologías dan la certeza de poder implementar, aunque no sea fácil, cualquier algoritmo.

Teniendo en cuenta los resultados del proyecto: “Sistema portátil para asistir a personas invidentes en navegación con objetivo en tiempo real” (Díaz et al., 2019), que fueron muy relevantes, se recomienda tener en cuenta los alcances de este estudio para terminar el módulo de planificación de trayectorias que hace falta en el dispositivo.

Se recomienda el uso de la programación en GPU (*Graphical Processing Unit*) que acelera los cálculos porque contienen múltiples unidades de procesamiento paralelo que son capaces de dar un rendimiento mucho mayor en la implementación de algoritmos que las CPU (*Central Processing Unit*)

Referencias Bibliográficas

Abellan, M, (2019). *Programación y electrónica de sensores básicos utilizando salidas digitales con Arduino*. https://www.programoergosum.es/

Alcalde, A. (2017). *Algoritmos de caminos cortos.* https://elbauldelprogramador.com/algoritmos-de-caminos-cortos/#:~:text=El%20algoritmo%20de%20Dijkstra%20consiste,grafo%2C%20el%20algoritmo%20se%20detiene.

Aragón, I, (2010) *Planificador orientativo inteligente de rutas e itinerarios turísticos en el metro metropolitano.* https://poiritem.wordpress.com/

Bueno, J. C. R., y Mantilla, K. J. G. (2020). *Sistemas embebidos y Hardware libre.* http://wiki.sc3.uis.edu.co/images/e/e1/GR7.pdf

Carranza, A. (2021). *Conoce qué es Java*. *crehana.* https://www.crehana.com/co/blog/desarrollo-web/que-es-java/

Chen, Q., Khan, M., Tsangouri, C., Yang, C., Li, B., Xiao, J., y Zhu, Z. (2017). *CCNY smart cane.* In 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER) (pp. 1246-1251). IEEE.

Cuevas, M (2021) *Ashirase de Honda Motor Co, desarrolla sistema de navegación para personas con discapacidad visual*. https://geeksroom.com/

Drozdek, A. (2007). *Estructura de datos y algoritmos en Java*. Ediciones Paraninfo

Dussan, C. (2020) *El censo de discapacidad y la Covid-19.* https://www.inci.gov.co/blog/el-censo-de-discapacidad-y-la-covid-19

Eigner S. (2021). *From the Foot Perspective: TU Graz Develops Algorithm for Shoe-Based Blind Assistance System*. Universidad Tecnológica de Graz. https://www.tugraz.at/

Eloviparo. (2023). *La distancia Manhattan o la distancia euclidea.* https://eloviparo.wordpress.com/2018/03/13/la-distancia-manhattan-o-la-distancia-euclidea/

Europa Press. (2014). *La discapacidad visual afecta a 285 millones de personas en el mundo*. https://www.europapress.es/epsocial/igualdad/noticia

Ferrís, R., y Albert,J.(2008). *Introducción al estudio de algoritmos y su complejidad*. https://informatica.uv.es/iiguia/AED/teoria/apuntes/cuatr2/AED.Tema.09.pdf

Gordillo, N (2007). Metodología, método y propuestas metodológicas en Trabajo Social. *Revista Tendencia & Retos*, 12, 119-135.

Hosting Plus. (2021). *Cómo funciona el lenguaje de programación Python*. https://www.hostingplus.com.co/blog/como-funciona-el-lenguaje-de-programacion-python/?gclid=EAIaIQobChMI3Yq-rdL69gIVkYjICh3jLwnREAAYASAAEgIPfvD_BwE

Hosting Plus. (2022). L*enguaje C++: cuáles son sus ventajas al programar*. https://www.hostingplus.com.co/blog/lenguaje-c-cuales-son-sus-ventajas-al-programar/?gclid=EAIaIQobChMI_tfp7c_69gIV2PrICh3lVQP2EAAYASAAEgIb3_D_BwE

https://doi.org /10.1109/I2CT.2018.8529757

Instituto Tecnológico de Nuevo Laredo. (2005). *Inteligencia artificial*. http://www.itnuevolaredo.edu.mx/takeyas/apuntes/Inteligencia%20Artificial/Apuntes/tareas_alumnos/A-Star/A-Star(2005-II-A).pdf

Islam, M., Raj, M., Nath, S., Rahman, M., Hossen, S., y Imam, M. (2018) An Indoor Navigation System For Visually Impaired People Using a Path Finding Algorithm and a Wearable Cap. *IEEE 3rd International Conference for Convergence in Technology*

Joyanes, L. (2006). *Programación en C++: Algoritmos, Estructuras de datos y objetos. http://biblioteca.univalle.edu.ni/files/original/ec89128132c7434f1765ceb9cbf1160828ae85f5.pdf*

Kelly. A, (2017) Motion-planning. *Mobile Robotics Mathematics, Models, and Methods* 1,. 640 – 690 DOI: https://doi.org/10.1017/CBO9781139381284

Koenig, S., y Likhachev, M. (2002). D*Lite. *Proceedings of the National Conference on Artificial Intelligence*

Laguna Sánchez, G. A., Olguın Carbajal, M., y Barrón Fernández, R. (2011). Introducción a la programación de códigos paralelos con CUDA y su ejecución en un GPU multi-hilos. *ContactoS*, 80, 65-69.

Lee, Y., y Medioni, G. (2015) *Wearable RGBD Indoor Navigation System for the Blind.* Conference: European Conference on Computer Vision. *DOI: https://doi.org/*10.1007/978-3-319-16199-0_35

Li B., Muñoz J.P., Rong X., Xiao J., Tian Y., y Arditi A. (2016). *ISANA: Wearable Context-Aware Indoor Assistive Navigation with Obstacle Avoidance for the Blind.* https://doi.org/10.1007/978-3-319-48881-3_31

Maikel, O., Torres Piñeiro, V., y Moreno Vega, A. (2009). *Study of two path planning methods in static environments.* RCCI, 3(1-2), 35-40

McGraw-Hill.

Mennecke, B. E., y Martin, D. C.(1996). *Sistemas de información geográfica: aplicaciones y oportunidades de investigación para investigadores de sistemas de información.* Conferencia internacional de Hawái sobre ciencias de sistemas, 1996.

Orellana, J.D. (2015). *Planificación de Trayectorias con Optimización Multi-Objetivo considerando modelos realistas de aeronaves.* Universidad de Sevilla

Pascual Estapé, J.A. (2021). *Inteligencia artificial: qué es, cómo funciona y para qué se utiliza en la actualidad. Computerhoy*. https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917

Patel, A. (2023). *Red Blob Games*. https://www.redblobgames.com/

Rendon Tejedor, B. (2021). *Big Data. Mailjet*. https://es.mailjet.com/blog/news/big-data/

Represa Pérez, C., Cámara Nebreda, J.M., y Sánchez Ortega. P.L. (2016). *Introducción a la programación en CUDA.* Universidad De Burgos.

RRP Redacción. (2020). *Cómo Google Maps predice el tráfico y determina las rutas que le pides*. https://rpp.pe/tecnologia/mas-tecnologia

Sanahuja, G., Valera, A., Sánchez, A. J., Ricolfe Viala, C., Vallés, M., y Marín, L. (2011). Control embebido de robots móviles con recursos limitados basado en flujo óptico. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 8(3), 250-257., https://doi.org/10.1016/j.riai.2011.06.012.

The MathWorks. (2022). *Descripción del producto MATLAB. MathWorks.* https://es.mathworks.com/help/matlab/learn_matlab/product-description.html

Tinetti, F., Martin, S., Frati, F., y Méndez, M (2013). Experiencias de optimización y paralelización mediante contadores de rendimiento de hardware. *Conferencia Internacional de Supercomputación México.*

Villarroel Salcedo, J. (2014). *Sistemas en Tiempo Real.* Centro Politécnico Superior, Universidad de Zaragoza

Yandún, A., y Sotomayor, N. (2017) *Planeación y seguimiento de trayectorias para un robot móvil MSc*. Escuela Politécnica Nacional, Quito – Ecuador.

Zhang H y Ye, C., (2019). Human-robot interaction for assisted wayfinding of a robotic navigation aid for the blind. *in International Conference on Human System Interaction, HSI, Jun. 2019, vol. 2019-June, pp. 137–142*, doi: 10.1109/HSI47298.2019.8942612.

Apéndice

Apéndice A

Enlace

Ver anexos en el correspondiente repositorio de Github

https://github.com/paulamortega/codigo.git En donde se encontrará los códigos de los algoritmos implementados. En entornos estáticos 5 algoritmos: *Wander Planner, Spanning tree, Breadfirst, Dijkatra y A start*. En entornos dinámicos 2 algoritmos D start y *A star From Scratch*.