

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“SISTEMA DE POSICIONAMIENTO INTERNO PARA LA
DETECCIÓN DE USUARIOS, REPRESENTACIÓN DE
PUNTOS DE INTERÉS, Y OPTIMIZACIÓN DE RUTAS
DENTRO DE ESPACIOS CERRADOS”

CRISTÓBAL IGNACIO ALVAREZ GONZALEZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Felipe Ignacio Beroiza Osses
Profesor Correferente: Marcello Visconti

Agosto - 2024

DEDICATORIA

A mi familia, amigos y a todos mis seres queridos que me han
acompañado en la vida.

AGRADECIMIENTOS

Inicialmente, quiero agradecer a mis padres y hermana por su apoyo en estos años de estudios, por cada sacrificio que tuvieron que hacer para permitirme estudiar esta carrera sin preocupaciones. Han sido fundamentales en este camino que ha tenido altos y bajos, y que por fin ha llegado a su fin.

También agradecer a mis amigos y compañeros de universidad Cristian, Emmanuel, Matías, Nicolás Pujante y Nicolás Puente, con los que he compartido muchísimas experiencias las cuales recordaré por siempre.

Agradezco también al profesor Felipe Beroiza por su disposición y apoyo a lo largo de este proceso de memoria. Su guía y respaldo fueron esenciales para superar los desafíos y finalizar este trabajo con éxito.

Finalmente, agradecer a todos mis seres queridos que me han acompañado a lo largo de este proceso, brindándome su apoyo, comprensión y compañía.

RESUMEN

Resumen— Los sistemas de posicionamiento interno son herramientas valiosas para obtener información sobre visitantes en espacios cerrados, permitiendo conseguir la localización sin interacción del usuario ni uso de GPS. Este trabajo presenta el desarrollo de un sistema que integra tecnología Wi-Fi de Cisco Meraki con algoritmos avanzados de procesamiento de datos y generación de rutas. Se implementó una aplicación móvil con interfaz 3D para visualizar mapas, rutas optimizadas y la posición del usuario en tiempo real, mejorando la experiencia de navegación en interiores y demostrando la viabilidad de soluciones localización en espacios comerciales cerrados.

Palabras Clave— Geolocalización, Generación de rutas, Aplicación móvil, Indoor Location, Puntos de interés

ABSTRACT

Abstract— Indoor positioning systems are valuable tools for obtaining information about visitors in enclosed spaces, allowing localization without user interaction or GPS use. This work presents the development of a system that integrates Cisco Meraki Wi-Fi technology with advanced data processing and route generation algorithms. A mobile application with a 3D interface was implemented to visualize maps, optimized routes, and real-time user position, enhancing the indoor navigation experience and demonstrating the feasibility of localization solutions in enclosed commercial spaces.

Keywords— Geolocalization, Routes Generation, App Mobile, Indoor Location, Points of Interest

GLOSARIO

GPS: Sistema de Posicionamiento Global o Global Positioning System.

AP: Punto de Acceso en una red Wi-Fi.

IEEE: Ingenieros Eléctricos y Electrónicos.

Wi-Fi: Wireless Fidelity, nombre para referirse a las redes inalámbricas.

BLE: Bluetooth Low Energy.

RFID: Identificación por radiofrecuencias.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	XII
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	2
1.1 Contexto y situación actual	2
1.2 Propuesta de solución	4
1.3 Objetivos	4
1.3.1 Objetivo General	4
1.3.2 Objetivos Específicos	5
1.4 Estructura de la Tesis	5
CAPÍTULO 2: Marco conceptual	6
2.1 Sistema de Posicionamiento Global	6
2.2 Factores que Afectan la Precisión del GPS	7
2.2.1 Difracción y Propagación Multicaminos	7
2.2.2 Obstrucción	7
2.2.3 Refracción	8
2.2.4 Interferencia	8
2.3 Tecnologías y sistemas para posicionamiento en interiores	8
2.3.1 Sistemas basados en Wi-Fi (WPS)	9
2.3.2 Bluetooth, BLE y Beacons	10
2.3.3 Identificación por Radiofrecuencia (RFID)	10
2.3.4 Ultra-Wideband	11
2.3.5 Li-Fi	11
2.3.6 Cisco Meraki	11
2.4 Conversión de señales inalámbricas a distancia	12
2.4.1 RSSI (Received Signal Strength Indicator)	12
2.4.2 ToF (Time of Flight)	13
2.4.3 TDoA (Time Difference of Arrival)	13
2.4.4 AoA (Angle of Arrival)	13
2.5 Nodos y búsqueda heurística	13
2.5.1 Definición de nodos	13
2.5.2 Búsqueda heurística	13

2.6	Modelos matemáticos para calcular posiciones	14
2.6.1	Triangulación	14
2.6.2	Trilateración	14
2.6.3	Multilateración	15
2.6.4	Filtro de Kalman	15
2.7	Transformación entre Coordenadas de Píxeles y Geográficas	17
2.8	Algoritmos y Métodos de Generación de Rutas	18
2.8.1	Algoritmos Offline	18
2.8.2	Algoritmo de Dijkstra	19
2.8.3	Algoritmo A* (A estrella)	19
2.8.4	Funciones Heurísticas en A*	20
2.8.5	Comparación entre Dijkstra y A*	20
2.8.6	Algoritmos Online	22
2.8.7	Algoritmo D* (D estrella)	22
2.8.8	RTA* (Real-Time A*)	25
2.8.9	LRTA* (Learning Real-Time A*)	25
2.8.10	Comparación algoritmos	26
2.9	Optimización de rutas	26
2.9.1	Algoritmo de Ramer-Douglas-Peucker	27
2.9.2	Algoritmo de Visvalingam-Whyatt	28
2.9.3	Algoritmo de Lang	29
2.10	Imperfecciones en rutas generadas por algoritmos de generación de rutas	29
2.10.1	Zigzag y “Dientes de sierra”	29
2.10.2	Oscilaciones y adherencia a los bordes	29
2.10.3	Cambios bruscos de dirección y falta de suavizado	29
2.11	Procesamiento de Imágenes	30
2.11.1	Técnicas de Procesamiento de Imágenes	30
2.11.2	Tecnologías y Herramientas	31
2.12	Aplicación Móvil	31
2.13	Estado del Arte	32
2.13.1	Comparación de Tecnologías de Posicionamiento en Interiores	32
2.13.2	Google Maps Indoor Location	34
2.13.3	Apple Maps Indoor Positioning	35
2.13.4	Sistema IndoorAtlas	37
2.13.5	Proyecto IPIN (Indoor Positioning and Indoor Navigation)	37
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN		39
3.1	Metodología utilizada	39
3.2	Generación de Mapa	40
3.3	Generación de Rutas	44
3.3.1	Optimización de Rutas	47
3.4	Aplicación Móvil	54
3.4.1	Tecnologías Utilizadas	55
3.4.2	Interfaz de Usuario	55

3.5	Servidor de Backend	58
3.5.1	Conexión con Cisco Meraki	59
3.5.2	Obtención de Ubicación	59
3.5.3	Obtención de rutas	60
3.6	Aproximación de Ubicación en Tiempo Real	60
3.6.1	Teoría del Filtro de Kalman	61
3.6.2	Implementación	62
3.6.3	Ajuste de Parámetros	63
3.6.4	Ajuste a Restricciones Físicas	63
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		65
4.1	Evaluación de algoritmos de rutas	65
4.1.1	Comparación técnica de algoritmos	66
4.1.2	Comparación visual de algoritmos	68
4.2	Filtrado de Kalman	70
CAPÍTULO 5: CONCLUSIONES		75
5.1	Trabajo futuro	76
ANEXOS		77
REFERENCIAS BIBLIOGRÁFICAS		78

ÍNDICE DE FIGURAS

1	Implementación de ubicación interna en Madison Square Garden.	4
2	Diagrama de la disposición de los satélites que componen el GPS.	6
3	Errores GPS.	8
4	Diagrama del funcionamiento de la API de Location Analytics de Cisco Meraki. .	12
5	Representación triangulación.	14
6	Representación de multilateración	15
7	Ejemplo de aplicación del Filtro de Kalman en la estimación de trayectoria. . .	17
8	Representación proceso de algoritmo de Ramer–Douglas–Peucker.	27
9	Representación algoritmo de Visvalingam–Whyatt.	28
10	Interfaz de Google Maps en un dispositivo móvil.	34
11	Interfaz de Apple Maps en un dispositivo móvil.	35
12	Diagrama de flujo compacto del proceso de desarrollo	40
13	Plano original espacio de prueba.	41
14	Mapa procesado manualmente, en donde el rojo indica las tiendas, amarillo las zonas transitables, cyan los estacionamientos, negro las paredes y blanco las zonas no identificadas.	42
15	Mapa procesado en donde las zonas azules corresponden a las zonas transitables y los puntos verdes a el centro de cada tienda.	44
16	Ruta generada (en azul) utilizando el algoritmo A* con distancia euclidiana como función heurística	46
17	Ruta generada (en azul) utilizando el algoritmo A* con distancia euclidiana como función heurística	47
18	Ruta generada (en azul) utilizando el algoritmo A* con distancia de Chebyshev como función heurística	48
19	Representación algoritmo de Ramer–Douglas–Peucker	50

20	Comparación de rutas optimizadas utilizando el algoritmo de Ramer–Douglas–Peucker con diferentes valores de ϵ . De izquierda a derecha: (a) $\epsilon = 0,5[m]$, resultando en 39 nodos y una ruta muy detallada; (b) $\epsilon = 5[m]$, produciendo 8 nodos y una simplificación moderada; (c) $\epsilon = 10[m]$, generando 6 nodos y una simplificación más agresiva. Se observa cómo el incremento de ϵ reduce la complejidad de la ruta a costa de perder algunos detalles de la trayectoria original.	51
21	Ruta generada (en azul) utilizando algoritmo A* y distancia Chebyshev como función heurística.	52
22	Ruta generada (en azul) con algoritmo A* y optimizada con algoritmo de Ramer–Douglas–Peucker	53
23	Visualización de mapa 3D utilizando contornos y Three.JS.	56
24	Visualización de tienda e información relevante.	57
25	Visualización de ruta utilizando Three.JS.	58
26	Diagrama del funcionamiento de la API de Location Analytics de Cisco Meraki. .	59
27	Diagrama funcionamiento de obtención de rutas.	60
28	Ruta original (en amarillo) junto a la ruta calculada por la red de Cisco Meraki (en multicolor).	61
29	Comparación tiempo de ejecución promedio.	66
30	Comparación tiempo de ejecución por iteración.	67
31	Gráfico de tiempo de ejecución promedio en función de la longitud de la ruta. .	68
32	Rutas extensas generadas con Algoritmo A* (Izquierda) versus Algoritmo de Dijkstra (Derecha), ambas simplificadas aplicando algoritmo de Ramer–Douglas–Peucker.	69
33	Rutas con obstáculos utilizando Algoritmo A* (Izquierda) versus Algoritmo de Dijkstra (Derecha), ambas simplificadas aplicando algoritmo de Ramer–Douglas–Peucker.	70
34	Ubicación de un dispositivo calculada por Cisco Meraki (Multicolor) y ruta real (Amarillo)	72
35	Posiciones predichas utilizando filtro de Kalman (Multicolor) y posiciones reales (Amarillo).	73

36 Errores de posición.	74
37 Resumen de enfoques tecnológicos y precisiones correspondientes para la investigación y tecnologías IPS disponibles comercialmente.	77

ÍNDICE DE TABLAS

1 Comparativa de métricas de distancia	22
2 Comparación entre los algoritmos A* y Dijkstra	23
3 Comparación entre los algoritmos D*, RTA* y LRTA*	26
4 Resumen de enfoques tecnológicos y precisiones correspondientes para la investigación y tecnologías IPS disponibles comercialmente.	33
5 Comparación de algoritmos de optimización de rutas	54
6 Comparación de Algoritmos de Ruta, donde V son el numero de vértices y E el numero de aristas del grafo.	65

INTRODUCCIÓN

Los avances en tecnología móvil y la creciente automatización de servicios han incrementado la demanda de soluciones de navegación, no solo en exteriores sino también en entornos cerrados [Mordor Intelligence, 2024]. Espacios públicos como centros comerciales, hospitales y universidades se han vuelto más complejos, presentando desafíos únicos para el desplazamiento eficiente de los clientes. En este contexto, los sistemas de posicionamiento en interiores han emergido como una herramienta útil para abordar estas necesidades, prometiendo mejorar la experiencia del usuario y optimizar la gestión de espacios [Basiri *et al.*, 2017].

Mientras que el Sistema de Posicionamiento Global (GPS) ha revolucionado la navegación en exteriores, su eficacia se suele ver severamente limitada en entornos cerrados debido a limitaciones físicas en su funcionamiento. Esta limitación ha impulsado el desarrollo de tecnologías alternativas para el posicionamiento en interiores, aprovechando infraestructuras existentes como redes Wi-Fi y Bluetooth [Gu *et al.*, 2009].

La presente memoria se centra en el desarrollo de un sistema de posicionamiento interno que une tecnologías de posición interna, algoritmos de procesamiento de datos y generación de rutas. El objetivo principal es proporcionar una solución para la localización de usuarios, la representación de puntos de interés y la optimización de rutas dentro de espacios cerrados.

Este trabajo busca como objetivo desarrollar una solución enfocada en los desafíos específicos del posicionamiento en interiores, como la precisión en entornos complejos, la eficiencia en la generación de rutas y la integración con dispositivos móviles. Al combinar tecnologías Wi-Fi de Cisco Meraki con algoritmos de procesamiento avanzados, se intenta crear una solución que mejore la navegación en interiores enfocado en clientes.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

1.1. Contexto y situación actual

En la era digital actual, la capacidad para navegar con precisión y eficacia no solo se limita a los espacios exteriores urbanos o naturales, sino que se extiende cada vez más hacia los complejos interiores, como centros comerciales, universidades, y hospitales. A pesar de los avances significativos en tecnologías de localización global, como el GPS, estas herramientas frecuentemente fallan en proporcionar servicios de navegación y orientación dentro de espacios cerrados, donde las señales satelitales son inaccesibles o ineficaces [Gu *et al.*, 2009]. Este déficit en la capacidad de localización en interiores ha impulsado la necesidad de desarrollar soluciones específicas que aborden los retos únicos presentados por estos entornos, como la densidad de obstáculos, la variedad de estructuras internas y la diversidad de necesidades de los usuarios [Yassin *et al.*, 2017]. A medida que los espacios interiores se vuelven más complejos y multifuncionales, la demanda de tecnologías de navegación más precisas y adaptativas es más necesaria que nunca, remarcando la relevancia de explorar nuevas aproximaciones y soluciones en este campo en expansión [Gu *et al.*, 2009].

Actualmente, se utiliza de forma masiva el GPS o *Global Positioning System*, creado por el Departamento de Defensa de Estados Unidos y liberado para uso civil desde la década de los noventas. Si bien esta tecnología funciona de manera óptima en zonas abiertas, tanto en ciudades como en zonas apartadas, en espacios más concurridos o cerrados los resultados son distintos. El GPS es un sistema que se compone de una constelación de 32 satélites, los cuales están constantemente enviando señales hacia la tierra [GPS.gov, 2024]. Debido a esta mecánica, las señales emitidas por los satélites pueden sufrir varios fenómenos. Entre los más comunes se encuentran:

- Interferencia y bloqueo: La señal emitida es bloqueada por edificios, estructuras o árboles. Además, al entrar a zonas ocultas, como túneles o edificios, la señal puede estar totalmente bloqueada.
- Errores de multicamino: Fenómeno que ocurre cuando, debido a la reflexión de la señal emitida, esta se retrasa, provocando cálculos erróneos en el dispositivo receptor.

Debido a estas problemáticas asociadas al GPS en espacios interiores, se buscaron otras alternativas a este sistema, las cuales sean compatibles con la georreferenciación en espacios interiores. Entre estas opciones, las principales y más usadas son:

- Wi-Fi: Sistema basado en la detección de la ubicación de un dispositivo basado en los

puntos de accesos conectados. Es una de las tecnologías más comunes debido a su fácil implementación en redes preexistentes.

- Bluetooth y Beacons: Sistema basado en la utilización de pequeños dispositivos llamados Beacons, los cuales emiten señales bajo el estándar Bluetooth Low Energy.

Sin embargo, la utilización de estas tecnologías está siendo cada vez más mermada debido a los estándares de privacidad que se están aplicando en el mundo tecnológico. Principalmente, los dispositivos de uso diario, como teléfonos inteligentes o tabletas, incorporan funcionalidades que permiten ocultar la dirección MAC de los dispositivos [Vanhoeft *et al.*, 2016]. Esta dirección es el equivalente al identificador de un dispositivo al comunicarse con la red, por lo que al realizar ciertos procesos de comunicación, la red recibe diferentes direcciones MAC del mismo dispositivo, imposibilitando la capacidad de ubicarlo en un espacio.

En este contexto, la solución propuesta en este proyecto utiliza redes Wi-Fi implementadas con tecnología de Cisco Meraki, una empresa que ofrece soluciones de red administrables desde una plataforma web en la nube. Entre sus funcionalidades se encuentra Location Analytics, la cual permite obtener la posición de dispositivos cercanos aprovechando la red Wi-Fi y los puntos de acceso de la red. Esta tecnología ofrece ventajas significativas sobre las alternativas mencionadas, ya que no depende exclusivamente de las direcciones MAC y proporciona información de ubicación precisa incluso cuando los dispositivos están conectados a la red Wi-Fi con una dirección IP asignada.

Por otro lado, aplicaciones de mapas y navegación ampliamente utilizadas, como Google Maps o Apple Maps, incluyen integraciones para la ubicación en espacios cerrados; sin embargo, estas no son comunes y generalmente solo están disponibles en lugares de gran afluencia mundial, como el Madison Square Garden o el aeropuerto JFK, ambos en Nueva York ¹. El enfoque de estas aplicaciones es principalmente sobre espacios públicos a gran escala, como ciudades o barrios, y no se enfocan en edificios en particular, como centros comerciales o hospitales, privándonos así de información valiosa para orientarnos dentro de estos espacios.

¹Esta afirmación se basa en una serie de verificaciones realizadas en junio de 2024, utilizando las aplicaciones Google Maps y Apple Maps en varios lugares de interés en diferentes ciudades a nivel mundial, incluyendo Santiago de Chile. Para ejemplos específicos de Google Maps (ya que Apple Maps no tiene versión web), consulte los siguientes links:

1. Google Maps: Costanera Center
2. Google Maps: Madison Square Garden
3. Google Maps: Estadio Maracanazinho

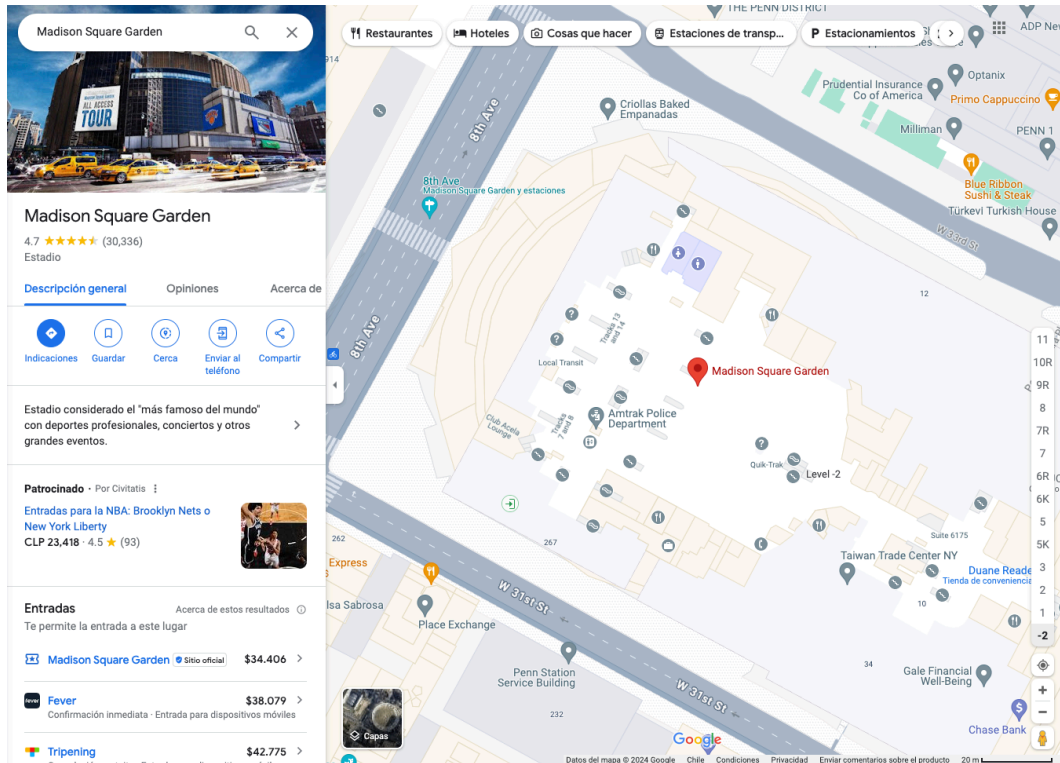


Figura 1: Implementación de ubicación interna en Madison Square Garden.

Fuente: [Google Maps, 2024b]

1.2. Propuesta de solución

Dado la situación anterior, las problemáticas y limitaciones que existen en el contexto de la navegación, representación de puntos de interés y obtención de ubicación en interiores, se propone desarrollar una aplicación móvil que represente espacios en 3D, en conjunto a un sistema de backend que, dado los dispositivos detectados por redes Wi-Fi basadas en Cisco Meraki, permita obtener tanto la ubicación en tiempo real del dispositivo como rutas óptimas hasta puntos de interés.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar una aplicación de navegación en interiores con redes Wi-Fi basadas en Cisco Meraki para mejorar la orientación y movilidad dentro de espacios cerrados. La aplicación proporciona a los usuarios la opción de obtener la posición actual y rutas optimizadas en un

entorno tridimensional.

1.3.2. Objetivos Específicos

- Procesar el plano del espacio físico de estudio y obtener una representación en forma de grafo.
- Generar rutas eficientes dentro de un espacio cerrado.
- Desarrollar un sistema para obtener la ubicación actual de dispositivos mediante la utilización de redes Wi-Fi de Cisco Meraki y su funcionalidad de localización interna.
- Integrar el modelado 3D con los datos de ubicación en tiempo real.
- Desarrollar una aplicación móvil que ofrece visualización de rutas óptimas y obtención de ubicación en tiempo real dentro de un espacio cerrado.

1.4. Estructura de la Tesis

Esta tesis está organizada de la siguiente manera:

- En el Capítulo 2, se introducirán los conceptos teóricos necesarios para desarrollar la solución, incluyendo los fundamentos de los modelos matemáticos, sistemas de posicionamiento y los algoritmos relevantes. Además, se estudiará el estado del arte, explorando las soluciones tanto comerciales como científicas que intentan dar solución a esta problemática.
- El Capítulo 3 presenta la propuesta de solución, detallando el paso a paso del desarrollo e implementación de la aplicación, enfocándose principalmente en el procesamiento de mapas, la generación de rutas y la implementación del filtro de Kalman.
- En el Capítulo 4 se expone la validación de la solución, presentando los resultados de las pruebas aplicadas y evaluando el rendimiento del sistema.
- Finalmente, en el Capítulo 5 se ofrecen las conclusiones del trabajo realizado y se sugieren posibles trabajos de investigación futura.

CAPÍTULO 2

Marco conceptual

A continuación se desarrollará el marco conceptual, enfocado en el fundamento teórico para el desarrollo de esta memoria.

2.1. Sistema de Posicionamiento Global

El GPS o Sistema de Posicionamiento Global, es un sistema desarrollado por el gobierno de Estados Unidos de América durante el siglo XX. Este sistema está compuesto por 32 satélites operativos. Estos satélites están distribuidos en 6 niveles equidistantes, lo cual permite que al menos 4 satélites tengan visibilidad directa sobre un dispositivo en cualquier lugar en la Tierra [GPS.gov, 2024]. Los satélites que componen el GPS emiten una señal continua hacia la tierra, la cual contiene información como la ubicación del satélite y el momento exacto en que fue emitida. Estas ondas son recibidas por los dispositivos en tierra, los cuales contrastando la información enviada junto al momento en que fue recibida, pueden aproximar la posición del dispositivo en la tierra.

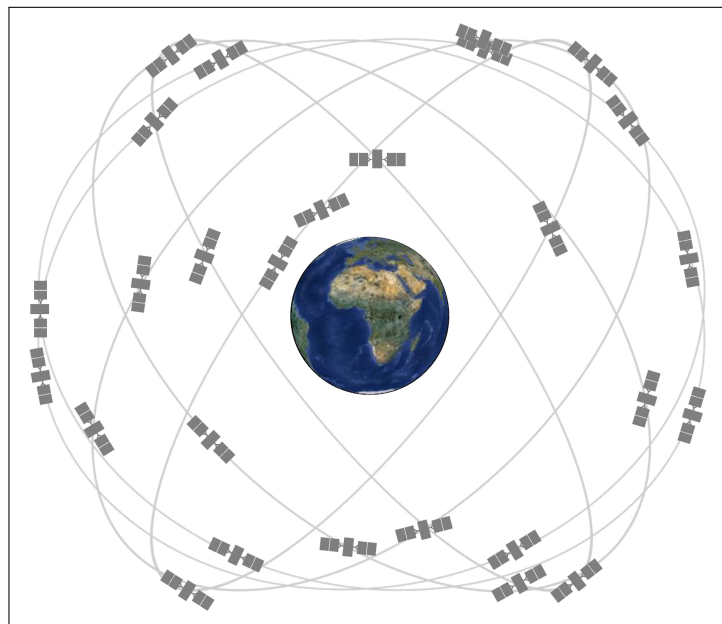


Figura 2: Diagrama de la disposición de los satélites que componen el GPS.

Fuente: [GPS.gov, 2024]

Esta constelación de satélites permite obtener la ubicación de un dispositivo en cualquier lugar de la Tierra con una precisión de metros (~ 15 m), mejorando sustancialmente en es-

pacios ampliamente despejados (~ 5 m) [Farjad *et al.*, 2019], en donde los principales casos de uso son:

- **Localización**, obtener la ubicación de un dispositivo en la Tierra. Por ejemplo, para expediciones a zonas remotas del planeta tierra, en donde es necesario saber la ubicación en todo momento.
- **Navegación**, obtener una ruta entre dos o varios puntos, como por ejemplo al usar aplicaciones de navegación como Google Maps o Waze al momento de conducir por la ciudad.
- **Seguimiento**, obtener la ubicación y estadísticas relacionadas a la posición de un dispositivo. Principalmente se aplica para el control de aviones o barcos de mercancías a lo largo de todo el globo.

2.2. Factores que Afectan la Precisión del GPS

2.2.1. Difracción y Propagación Multicaminos

Ocurre cuando una onda electromagnética emitida, en este caso, desde un satélite GPS es difractada debido a las condiciones del ambiente, como lo puede ser un bosque, montañas cercanas o construcciones de las grandes ciudades. Esto provoca que la misma señal emitida desde el satélite recorra una mayor distancia a la real, provocando que esta sea recibida por el receptor en un tiempo posterior al que realmente debería ser. Esta discrepancia en los tiempos medidos puede provocar un error al calcular la posición del dispositivo en la Tierra. [Kos *et al.*, 2010]

2.2.2. Obstrucción

Ocurre cuando la onda emitida desde el satélite perteneciente al GPS es bloqueada parcial o totalmente por elementos en el ambiente, como puede ser dentro de construcciones o en túneles. Esto provoca que las ondas emitidas por el satélite sean recibidas en un tiempo mucho mayor al real o que simplemente no lleguen hasta el receptor. Esto provoca que el GPS no pueda obtener una posición aproximada del dispositivo objetivo.

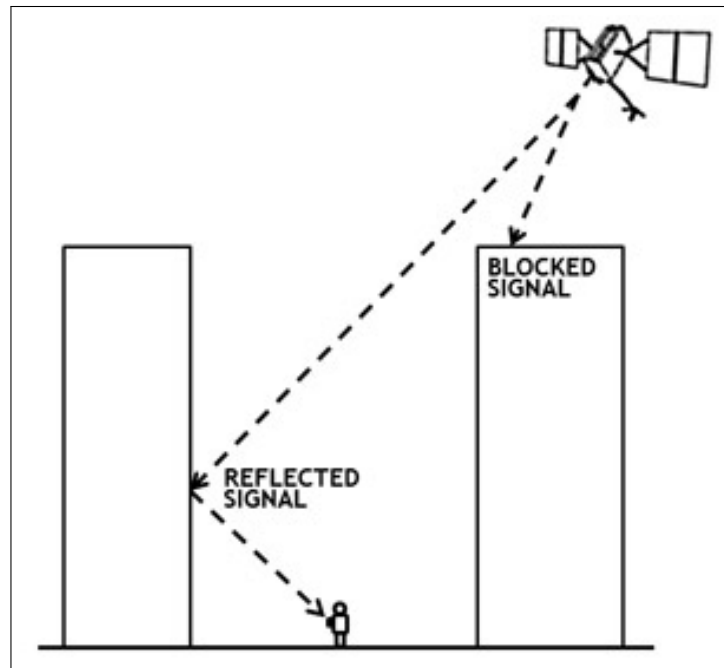


Figura 3: Errores GPS.
Fuente: [U.S. Government, 2023]

2.2.3. Refracción

La refracción es un fenómeno que ocurre cuando una onda cambia de dirección o velocidad al pasar de un medio a otro. Este fenómeno ocurre en diferentes escenarios en la tierra, por ejemplo al atravesar diferentes capas de la atmósfera o al atravesar fenómenos climáticos, como tormentas. En el contexto del GPS, puede causar errores al calcular de la distancia entre el satélite y el receptor.

2.2.4. Interferencia

La interferencia ocurre cuando una onda, en este caso una onda emitida por un satélite del GPS, se superpone con otra onda electromagnética ajena al sistema. Esto puede degradar la onda o incluso bloquearla por completo.

2.3. Tecnologías y sistemas para posicionamiento en interiores

Son sistemas complejos que se utilizan para ubicar dispositivos en espacios en donde el GPS no funciona de manera efectiva [Gu et al., 2009], entre los cuales se encuentran centros co-

merciales, hospitales o almacenes industriales, en donde obtener la posición puede significar disminuir costos en procesos o incluso salvar vidas.

Existen varios métodos para obtener la posición interna de un dispositivo en un momento dado, las cuales aprovechan distintos estándares y tecnologías utilizadas masivamente. Entre las más influyentes se encuentran [Zafari *et al.*, 2019].

2.3.1. Sistemas basados en Wi-Fi (WPS)

El Wi-Fi es un conjunto de estándares para comunicación inalámbrica que fue introducido por el Instituto de Ingenieros Eléctricos y Electrónicos (o IEEE por sus siglas en inglés) en el año 1997 bajo el nombre de estándar IEEE 802.11. El Wi-Fi permite que diferentes dispositivos electrónicos puedan conectarse a una red local cercana de manera inalámbrica utilizando ondas electromagnéticas como canal de comunicación [IEEE Standards Association, 2024]. Esta tecnología ha evolucionado a través de los años, comenzando con una velocidad de transmisión de datos de 2 Mbit/s en su primera versión, hasta llegar a velocidades cercanas a los 40 Gbit/s en su última versión. Además, el Wi-Fi permite una transmisión de datos de hasta 1 kilómetro a la redonda y múltiples puntos de acceso en la misma red. Si bien su uso principal es la comunicación, también se puede utilizar para otras aplicaciones como la localización en interiores.

Las redes Wi-Fi utilizan ondas electromagnéticas para comunicarse. Para implementar una red inalámbrica en espacios cerrados y complejos se debe cubrir el área deseada con emisores de estas ondas. Para lograr esta cobertura es necesario cubrir el área con puntos de accesos inalámbricos, los cuales usualmente están conectados de manera alámbrica a la red local y permiten que distintos dispositivos electrónicos se conecten a la red a través del Wi-Fi. Dependiendo de la configuración de las redes Wi-Fi y sus puntos de acceso, es posible que los dispositivos sean visibles por varios puntos de acceso de la misma red. De esta manera, varios puntos de acceso reciben información del mismo dispositivo pero con distintos tiempos de propagación entre el dispositivo y el receptor. De esta manera, y conociendo las posiciones de los puntos de acceso, es posible aplicar métodos geométricos para obtener la posición de un dispositivo en el plano.

Los dispositivos que se conectan a redes Wi-Fi, tales como dispositivos móviles o computadoras, envían periódicamente un frame (o paquete) llamado *Probe Request*, el cual se utiliza para descubrir redes Wi-Fi cercanas. Este frame contiene información como la dirección MAC e información técnica del dispositivo, como capacidades de red soportadas. Las *Probe Request* son sumamente utilizadas para detectar dispositivos de manera pasiva, ya que los puntos de acceso Wi-Fi y dispositivos en modo de escucha pueden recibir estos frames periódicamente sin la necesidad de realizar una acción en el dispositivo emisor. Por lo mismo, estos frames son sumamente utilizados para detectar y ubicar dispositivos en un espacio físico a través de múltiples puntos de acceso. Para minimizar los riesgos de privacidad, como el seguimiento no autorizado de la ubicación de los usuarios, los dispositivos actuales

cuentan con un sistema que aleatoriza su dirección MAC en cada *Probe Request*, limitando la capacidad de seguimiento de los dispositivos, aunque no es efectivo en todos los casos [Oliveira *et al.*, 2019].

2.3.2. Bluetooth, BLE y Beacons

Es un conjunto de estándares de comunicación entre dispositivos en movimiento. La última versión del Bluetooth permite una tasa de transmisión de datos de hasta 24Mbit/s y un rango de transmisión de hasta 100 metros. A diferencia del Wi-Fi, el Bluetooth tiene una tasa de transmisión inferior y no abarca una área tan extensa. Sin embargo, esta tecnología igualmente es utilizada para la localización de dispositivos en interiores, aunque con algunas diferencias. Mientras el Wi-Fi permite la conexión a la red y la localización mediante la triangulación y trilateración utilizando múltiples puntos de acceso, Bluetooth se basa más comúnmente en la proximidad a dispositivos conocidos como beacons, que emiten señales que los dispositivos pueden detectar y usar para determinar su ubicación relativa.

Bluetooth Low Energy (BLE) fue introducido en Bluetooth 4.0 y es una implementación del Bluetooth de bajo consumo. BLE ha mejorado significativamente el posicionamiento en interiores mediante el uso de beacons, los cuales son transmisores BLE que emiten señales periódicas con identificadores únicos, con un alcance de entre 10 a 30 metros. Para el posicionamiento, los beacons se suelen distribuir en el espacio y los dispositivos móviles estiman su distancia a ellos basándose en la intensidad de la señal recibida. Utilizando varios beacons, es posible determinar la posición de un dispositivo con una precisión de 1-3 metros [Zhuang *et al.*, 2016].

2.3.3. Identificación por Radiofrecuencia (RFID)

La Identificación por Radiofrecuencia (RFID) es una tecnología ampliamente utilizada en la localización en interiores. Un sistema RFID típico consta de etiquetas distribuidas en un espacio o adheridas a objetos/personas, y lectores ubicados en posiciones estratégicas. Para la localización en interiores se utilizan principalmente etiquetas activas (con batería) que emiten señales periódicamente; los lectores RFID captan estas señales y, mediante técnicas como la triangulación o el análisis de la intensidad de la señal recibida (RSSI), determinan la ubicación de la etiqueta. Esta tecnología es especialmente útil en entornos como almacenes, hospitales y oficinas grandes, donde ofrece ventajas como la capacidad de rastrear múltiples etiquetas simultáneamente, funcionar sin línea de visión directa y bajo consumo energético [Belhadi *et al.*, 2014].

2.3.4. Ultra-Wideband

Ultra-Wideband (UWB) es una tecnología de radiofrecuencia que utiliza señales de muy corta duración y amplio espectro para la localización en interiores. UWB ofrece una precisión alta comparada con otras tecnologías como Wi-Fi o Bluetooth, con errores de localización típicamente inferiores a 30 cm, debido a que logra medir con gran exactitud el tiempo de llegada de las señales. Los sistemas UWB no se ven afectados por fenómenos de interferencia y multicamino, lo que los hace ideales para entornos complejos. Además, su bajo consumo energético y su capacidad para atravesar paredes la convierten en una opción atractiva cuando se requiere alta precisión. Sin embargo, su implementación es costosa debido al hardware especializado [Alarifi *et al.*, 2016].

2.3.5. Li-Fi

Li-Fi (Light Fidelity) es una tecnología de comunicación inalámbrica que utiliza luz visible o infrarroja para transmitir datos, desarrollada como una alternativa a Wi-Fi. Li-Fi utiliza LEDs que se encienden y apagan a alta velocidad para enviar información, y fotodetectores para recibirla. Esta tecnología ofrece velocidades de transmisión potencialmente más altas que Wi-Fi, debido a las frecuencias en las que operan los LEDs. En el contexto del posicionamiento en interiores, Li-Fi utiliza las luces LED existentes, modificándolas para emitir identificadores únicos. Los dispositivos móviles equipados con fotodetectores captan estas señales de fuentes Li-Fi cercanas, y el sistema calcula la posición del dispositivo utilizando técnicas como triangulación o análisis de la intensidad de la señal, basándose en las señales recibidas. Esta tecnología puede ofrecer una precisión de hasta 10 cm en condiciones óptimas, superando a muchas otras tecnologías de posicionamiento en interiores y haciendo de Li-Fi una solución particularmente eficaz [Almadani *et al.*, 2019].

2.3.6. Cisco Meraki

Cisco Meraki es una empresa que ofrece soluciones de redes gestionadas desde la nube. Ofrecen productos desde puntos de acceso, routers y switches, hasta cámaras de seguridad y sensores, todos administrados desde su controlador en la nube, con el cual los dispositivos se sincronizan. Dentro de la gran cantidad de servicios que ofrece Cisco Meraki, se encuentra *Location Analytics*, el cual utiliza los puntos de acceso de una red para detectar, localizar y obtener estadísticas de los dispositivos cercanos a una red [Meraki, 2024].

Específicamente, Cisco Meraki utiliza la información obtenida cuando un dispositivo emite una *Probe Request* hasta uno o varios puntos de acceso de una red. Las *Probe Request* son paquetes enviados desde el dispositivo hacia una red Wi-Fi, los cuales contienen información del dispositivo, como su dirección MAC. Estos paquetes son enviados en intervalos regulares y son utilizados para descubrir redes Wi-Fi cercanas. Sin embargo, si uno de estos paquetes

es detectado por 3 o más puntos de acceso, es posible obtener su ubicación aproximada aplicando técnicas de triangulación.

Si bien Cisco Meraki no ofrece una aplicación ni una manera directa para que los clientes visualicen su posición en tiempo real dentro de espacios cerrados, es posible utilizarla para obtener la información de dispositivos sin la necesidad de implementar una solución propia. El principal beneficio de utilizar este servicio, es que cuenta con una API de uso público, la cual envía de forma periódica la información de los dispositivos detectado por la red, como se muestra en la figura.

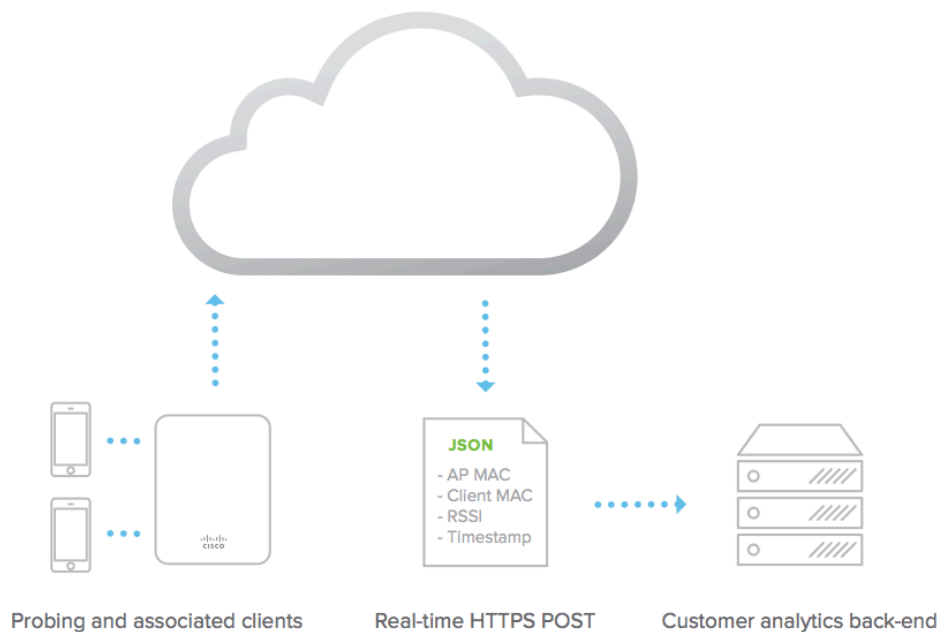


Figura 4: Diagrama del funcionamiento de la API de Location Analytics de Cisco Meraki.
Fuente: [Meraki, 2024].

2.4. Conversión de señales inalámbricas a distancia

La conversión de señales inalámbricas a distancia es un proceso importante en los sistemas de posicionamiento en interiores. Existen varios métodos para realizar esta conversión:

2.4.1. RSSI (Received Signal Strength Indicator)

El RSSI mide la potencia de una señal recibida. A medida que una señal viaja, su potencia disminuye. Utilizando modelos de propagación, es posible estimar la distancia entre el transmisor y el receptor basándose en la pérdida de potencia de la señal [Azmi *et al.*, 2018].

2.4.2. ToF (Time of Flight)

Este método mide el tiempo que tarda una señal en viajar desde el transmisor al receptor. Conociendo la velocidad de propagación de la señal (generalmente la velocidad de la luz para señales de radio), se puede calcular la distancia [Liu *et al.*, 2007].

2.4.3. TDoA (Time Difference of Arrival)

TDoA utiliza la diferencia en los tiempos de llegada de una señal a múltiples receptores para determinar la posición. Este método no requiere sincronización entre el transmisor y los receptores, lo que lo hace más robusto en ciertos escenarios [Azmi *et al.*, 2018].

2.4.4. AoA (Angle of Arrival)

AoA determina el ángulo desde el cual se recibe una señal. Combinando información de ángulo de múltiples receptores, se puede triangular la posición del transmisor [Liu *et al.*, 2007].

2.5. Nodos y búsqueda heurística

2.5.1. Definición de nodos

En el contexto de esta memoria y de los algoritmos de generación de rutas, un nodo corresponde a una posición específica en el espacio de búsqueda, los cuales pueden ser posiciones geográficas (latitud/longitud) o posiciones en una imagen (píxeles).

2.5.2. Búsqueda heurística

La búsqueda heurística es una técnica de resolución de problemas que utiliza una función heurística para estimar el costo o la distancia desde un estado actual hasta el objetivo. Esta técnica es especialmente útil en problemas donde el espacio de búsqueda es demasiado grande para una búsqueda exhaustiva, permitiendo encontrar soluciones de manera más eficiente en un tiempo acotado [Kokash, 2005]. La eficacia de la búsqueda heurística depende directamente de la función utilizada. Una buena función puede minimizar significativamente el número de nodos explorados, y por lo tanto, el tiempo de ejecución.

2.6. Modelos matemáticos para calcular posiciones

2.6.1. Triangulación

La triangulación es una técnica de localización que permite determinar la posición de un punto midiendo los ángulos desde este punto hacia al menos dos puntos de referencia con coordenadas conocidas. Se basa en las propiedades geométricas de los triángulos para calcular las coordenadas del punto objetivo. Este método utiliza la trigonometría para resolver el problema de posicionamiento [Edelsbrunner *et al.*, 1990].

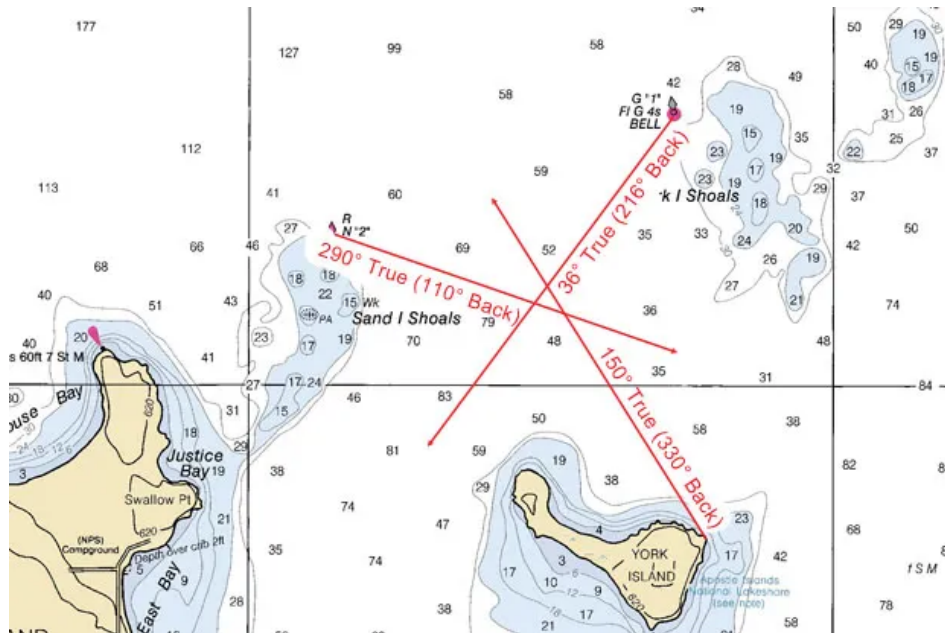


Figura 5: Representación triangulación.
Fuente: [Hansel, 2023].

2.6.2. Trilateración

La trilateración es una técnica de localización que determina la posición de un punto objetivo utilizando las distancias conocidas desde este punto a tres (o más) puntos de referencia con posición conocidas. Se basa en la intersección de esferas o circunferencias, cada una centrada en un punto de referencia y con radio igual a la distancia medida. Este método se aplica en sistemas como GPS y localización de dispositivos móviles [Rahman, 2012].

2.6.3. Multilateración

Multilateración es una técnica de localización que determina la posición de un objeto midiendo las diferencias en el tiempo de llegada de una señal emitida desde el objeto hacia múltiples receptores ubicados en posiciones conocidas. En lugar de medir distancias directas, la multilateración calcula estas diferencias de tiempo para formar hipérbolas en un espacio bidimensional o tridimensional, y la intersección de estas hipérbolas define la ubicación del objeto. Este método se utiliza en sistemas de navegación aérea y radares, así como en aplicaciones de posicionamiento en interiores utilizando señales de Wi-Fi o Bluetooth.

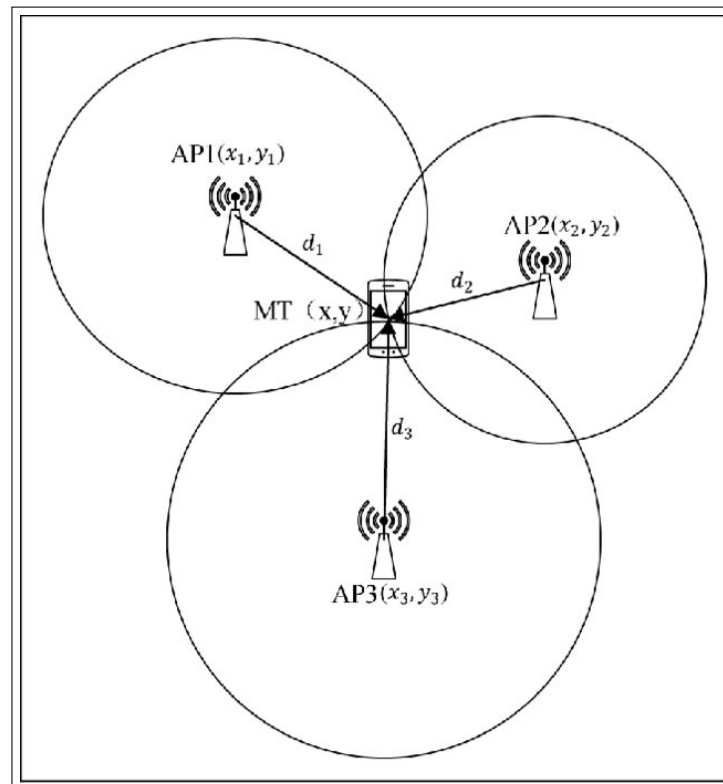


Figura 6: Representación de multilateración
Fuente: [Yang *et al.*, 2018]

2.6.4. Filtro de Kalman

El Filtro de Kalman, desarrollado por Rudolf E. Kalman en 1960, es un algoritmo recursivo que estima el estado óptimo de un sistema dinámico en tiempo real [Busu y Busu, 2021]. Su potencia radica en su capacidad para procesar mediciones ruidosas e incompletas, proporcionando estimaciones precisas del estado del sistema.

Funcionamiento del Filtro de Kalman El algoritmo opera en dos fases principales:

1. **Fase de Predicción:** En esta etapa, el filtro predice el estado futuro del sistema basándose en su estado actual y un modelo de cómo el sistema evoluciona en el tiempo. Esta predicción incluye:
 - Estimación del estado futuro
 - Cálculo de la incertidumbre asociada a esta predicción
2. **Fase de Actualización:** Aquí, el filtro refina su predicción incorporando nuevas mediciones. Este proceso implica:
 - Cálculo de la Ganancia de Kalman, que determina cuánto confiar en la nueva medición versus la predicción
 - Actualización de la estimación del estado
 - Actualización de la incertidumbre asociada a la nueva estimación

Aplicación en la localización en interiores En el contexto de la localización en interiores, el Filtro de Kalman se utiliza para:

- Fusionar datos de múltiples sensores (por ejemplo, WiFi, Bluetooth, acelerómetros)
- Reducir el ruido en las mediciones de posición
- Predecir la posición futura basada en la trayectoria actual y pasada
- Manejar interrupciones temporales en las señales de los sensores

La Figura 8 ilustra un ejemplo de aplicación del Filtro de Kalman, donde se puede observar cómo el algoritmo suaviza la trayectoria estimada, reduciendo el efecto del ruido en las mediciones.

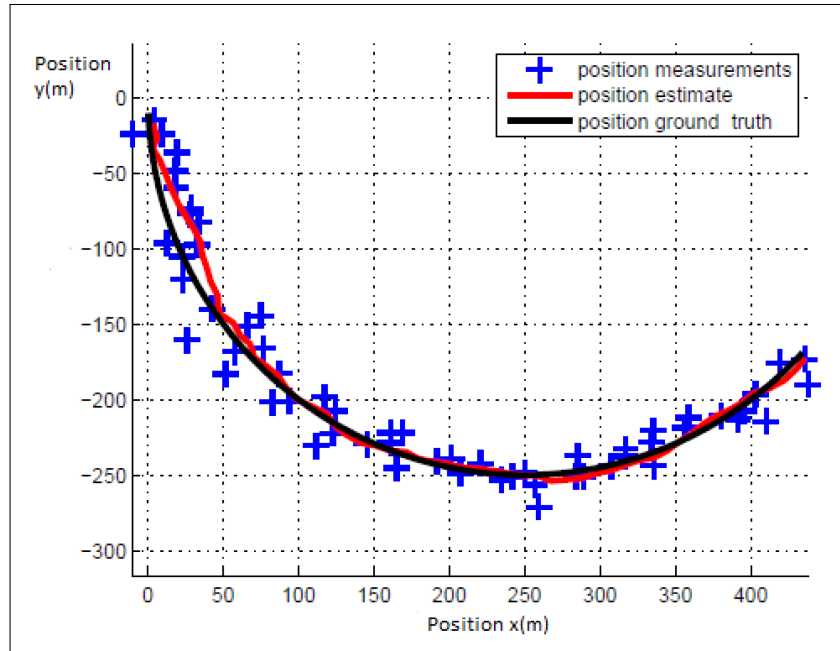


Figura 7: Ejemplo de aplicación del Filtro de Kalman en la estimación de trayectoria.
Fuente: [Busu y Busu, 2021]

La implementación del Filtro de Kalman en sistemas de localización en interiores permite obtener estimaciones de posición más estables y precisas, mejorando significativamente la experiencia del usuario en aplicaciones de navegación en interiores.

2.7. Transformación entre Coordenadas de Píxeles y Geográficas

Para transformar entre coordenadas de píxeles (x , y) y coordenadas geográficas (latitud, longitud), se utiliza un método de ajuste de plano. Este enfoque asume una relación lineal entre los dos sistemas de coordenadas, lo cual es una aproximación razonable para áreas geográficas relativamente pequeñas [Ghilani, 2017].

El proceso de transformación se realiza en los siguientes pasos:

1. **Ajuste de plano:** Se utiliza el método de mínimos cuadrados para ajustar un plano a los puntos de referencia conocidos. Para cada transformación (x , y a lat, lng y viceversa), se ajustan dos planos:

$$z = ax + by + c \quad (1)$$

donde z representa la coordenada que se desea obtener (x , y , latitud o longitud), y x e y son las coordenadas de entrada.

2. **Cálculo de coeficientes:** Los coeficientes a , b , y c se obtienen resolviendo el sistema de ecuaciones lineales:

$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix} \quad (2)$$

donde (x_i, y_i, z_i) son los puntos de referencia conocidos y n es el número total de puntos.

3. **Transformación de coordenadas:** Una vez obtenidos los coeficientes, se pueden transformar nuevos puntos utilizando la ecuación del plano:

$$z_{nuevo} = ax_{nuevo} + by_{nuevo} + c \quad (3)$$

4. **Interpolación:** Este método permite interpolar coordenadas desconocidas basándose en los puntos de referencia conocidos, proporcionando una estimación razonable para puntos dentro del área cubierta por los puntos de referencia.

Es importante notar que este método asume una relación lineal entre las coordenadas de píxeles y las coordenadas geográficas. Esta aproximación es generalmente válida para áreas geográficas pequeñas, pero puede perder precisión en áreas más grandes donde la curvatura de la Tierra se vuelve significativa.

2.8. Algoritmos y Métodos de Generación de Rutas

En el desarrollo de aplicaciones de navegación, la generación eficaz de rutas es fundamental para facilitar la orientación y el movimiento dentro de espacios físicos. Para esto, existen una serie de algoritmos que se enfocan en encontrar rutas óptimas entre dos o más puntos diferentes. [Rafiq *et al.*, 2020]

2.8.1. Algoritmos Offline

Los algoritmos de generación de rutas offline son conjuntos de algoritmos que tienen completo conocimiento del ambiente desde el comienzo de la ejecución, por lo que están mas capacitados para encontrar rutas completamente óptimas [Jaillet y Wagner, 2008].

2.8.2. Algoritmo de Dijkstra

Es un algoritmo creado en 1959 por su homónimo Edsger Dijkstra, el cual se enfoca en obtener la ruta más corta entre un nodo de inicio y todos los nodos restantes de un grafo, en donde las aristas tienen un peso o costo asociado. De manera simplificada, este algoritmo itera sobre los nodos vecinos, comenzando desde el nodo inicial. Si al visitar un nodo vecino el costo de la ruta disminuye, entonces actualiza la ruta conocida, y luego continúa hasta otro nodo vecino hasta que itera sobre todos los nodos o que los nodos que quedan por explorar tengan costo infinito [Madkour *et al.*, 2017].

Algorithm 1 Algoritmo de Dijkstra

```

1: procedure Dijkstra( $G, s$ )
2:    $dist[s] \leftarrow 0$  ▷ Distancia a nodo inicial
3:   for cada vértice  $v \in V[G] - \{s\}$  do
4:      $dist[v] \leftarrow \infty$ 
5:   end for
6:    $S \leftarrow \emptyset$  ▷ Conjunto de vértices visitados
7:    $Q \leftarrow V[G]$  ▷ Cola de prioridad con todos los vértices
8:   while  $Q \neq \emptyset$  do
9:      $u \leftarrow \text{ExtractMin}(Q)$  ▷ Vértice con menor distancia
10:     $S \leftarrow S \cup \{u\}$ 
11:    for cada vértice  $v$  adyacente a  $u$  do
12:      if  $dist[v] > dist[u] + w(u, v)$  then
13:         $dist[v] \leftarrow dist[u] + w(u, v)$ 
14:         $parent[v] \leftarrow u$ 
15:      end if
16:    end for
17:  end while return  $dist, parent$ 
18: end procedure

```

2.8.3. Algoritmo A* (A estrella)

Es una implementación mejorada del algoritmo de Dijkstra que se utiliza para encontrar la ruta menos costosa entre dos nodos de un grafo. A* se caracteriza por utilizar heurísticas según el contexto, para calcular la distancia (o costo) entre el nodo actual y el nodo objetivo. Este algoritmo comienza desde el nodo inicial y selecciona el vecino que no haya sido completamente explorado y que minimice la suma entre la distancia entre ellos, y la heurística aplicada al nodo vecino. De esta manera, el algoritmo va iterando sobre los nodos vecinos que no han sido completamente explorado y que minimice el costo total, hasta llegar al nodo objetivo.

2.8.4. Funciones Heurísticas en A*

Dentro de las funciones heurísticas mas utilizadas, están las siguientes:

- **Distancia Euclidiana:** Es la distancia en línea recta entre dos puntos en un espacio euclidiano. Para dos puntos en un plano bidimensional con coordenadas (x_1, y_1) y (x_2, y_2) , se calcula como:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Caso de uso: Es útil en espacios abiertos donde el movimiento no está restringido a direcciones específicas, como en la navegación de drones o en juegos con terreno abierto.

- **Distancia Chebyshev:** También conocida como distancia de tablero de ajedrez, es la distancia máxima entre las diferencias de las coordenadas. Para dos puntos (x_1, y_1) y (x_2, y_2) , se define como:

$$d = \max(|x_2 - x_1|, |y_2 - y_1|)$$

Caso de uso: Es ideal para situaciones donde el movimiento puede realizarse en todas las direcciones, incluyendo diagonales, con el mismo costo. Se utiliza comúnmente en juegos de estrategia por turnos o simulaciones donde las unidades pueden moverse en cualquier dirección en una cuadrícula.

- **Distancia Manhattan:** También llamada distancia de ciudad o norma L1, es la suma de las diferencias absolutas de las coordenadas. Para dos puntos (x_1, y_1) y (x_2, y_2) , se calcula como:

$$d = |x_2 - x_1| + |y_2 - y_1|$$

Caso de uso: Es particularmente útil en entornos donde el movimiento está restringido a direcciones ortogonales, como en la planificación de rutas en ciudades con estructura de cuadrícula o en la navegación interior de edificios con pasillos perpendiculares.

Este algoritmo es ampliamente utilizado de aplicaciones de navegación o incluso en videojuegos debido a la aplicación de heurísticas. A partir de distintas heurísticas es posible añadir restricciones que condicionen el comportamiento del algoritmo en entornos específicos. Esto permite una mayor flexibilidad y adaptabilidad, optimizando la eficiencia de la búsqueda según el contexto en que se encuentra el algoritmo.

2.8.5. Comparación entre Dijkstra y A*

Los algoritmos A* y Dijkstra se enfocan en encontrar el camino más corto en un grafo, sin embargo se diferencian en su enfoque y eficiencia. La principal diferencia es que el algoritmo A* utiliza una función heurística para estimar el costo desde el nodo actual hasta el

Algorithm 2 Algoritmo A* (A estrella)

```
1: procedure AStar(inicio, meta)
2:   abiertos  $\leftarrow$  {inicio}                                ▷ Conjunto de nodos por explorar
3:   cerrados  $\leftarrow$   $\emptyset$                                 ▷ Conjunto de nodos explorados
4:   g[inicio]  $\leftarrow$  0                                    ▷ Costo real desde el inicio
5:   f[inicio]  $\leftarrow$  g[inicio] + h(inicio, meta)        ▷ Función heurística
6:   while abiertos  $\neq$   $\emptyset$  do
7:     actual  $\leftarrow$  nodo en abiertos con menor f
8:     if actual = meta then return ReconstruirCamino(actual)
9:     end if
10:    abiertos.Eliminar(actual)
11:    cerrados.Agregar(actual)
12:    for cada vecino de actual do
13:      if vecino  $\in$  cerrados then
14:        continue
15:      end if
16:      costo_tentativo  $\leftarrow$  g[actual] + distancia(actual, vecino)
17:      if vecino  $\notin$  abiertos then
18:        abiertos.Agregar(vecino)
19:      else if costo_tentativo  $\geq$  g[vecino] then
20:        continue
21:      end if
22:      padre[vecino]  $\leftarrow$  actual
23:      g[vecino]  $\leftarrow$  costo_tentativo
24:      f[vecino]  $\leftarrow$  g[vecino] + h(vecino, meta)        ▷ Función heurística
25:    end for
26:  end while return fallo                                ▷ No se encontró camino
27: end procedure
```

Métrica	Característica principal	Tipo de movimiento	Caso de uso típico
Distancia Euclidiana	Distancia en línea recta entre dos puntos	Movimiento libre en cualquier dirección	Espacios abiertos, navegación de drones, juegos con terreno abierto
Distancia Chebyshev	Distancia máxima entre las diferencias de coordenadas	Movimiento en todas direcciones incluyendo diagonales con el mismo costo	Juegos de estrategia por turnos, movimiento en cuadrículas con diagonales permitidas
Distancia Manhattan	Suma de las diferencias absolutas de coordenadas	Movimiento restringido a direcciones ortogonales	Planificación de rutas en ciudades, navegación interior con pasillos perpendiculares

Tabla 1: Comparativa de métricas de distancia

objetivo, lo que permite priorizar la exploración de nodos que parecen minimizar la llegada al nodo final. Esta heurística permite la búsqueda de manera más informada, lo que generalmente resulta en una exploración más eficiente, especialmente en grafos grandes. Por otro lado, Dijkstra explora todos los nodos de manera uniforme, expandiéndose en todas las direcciones, lo que lo hace más completo pero potencialmente menos eficiente en ciertos escenarios. Finalmente, A* suele ser más óptimo que Dijkstra para encontrar caminos directos, especialmente cuando se aplica una función heurística adecuada al contexto. Dijkstra garantiza encontrar el camino más corto a todos los nodos del grafo, mientras que A* se enfoca en encontrar el camino más corto a un destino específico.

2.8.6. Algoritmos Online

Los algoritmos de generación de rutas online corresponden a un conjunto de algoritmos que no conocen todo el espacio desde el inicio y deben adaptarse tomando decisiones en tiempo real basándose en cambios, tales como obstáculos o cambios en el espacio[Jaillet y Wagner, 2008].

2.8.7. Algoritmo D* (D estrella)

Es un algoritmo de búsqueda de rutas óptimo y eficiente aplicado en entornos que van cambiando de manera dinámica, lo que lo convirtió en un algoritmo sumamente utilizado en el área de la robótica debido a su capacidad de adaptarse a ambientes dinámicos. Su

Característica	A*	Dijkstra
Enfoque principal	Encontrar el camino más corto a un destino específico	Encontrar el camino más corto a todos los nodos del grafo
Uso de heurística	Sí, utiliza una función heurística para estimar el costo al nodo objetivo	No utiliza heurística
Método de exploración	Prioriza nodos que parecen minimizar la llegada al nodo final	Explora todos los nodos de manera uniforme
Eficiencia en grafos grandes	Generalmente más eficiente	Potencialmente menos eficiente
Dirección de expansión	Dirigida hacia el objetivo	Se expande en todas las direcciones
Complejidad	Puede no explorar todo el grafo	Explora todo el grafo

Tabla 2: Comparación entre los algoritmos A* y Dijkstra

funcionamiento es similar a el algoritmo A*, ya que ambos utilizan funciones heurísticas para guiar la búsqueda hasta el objetivo final, sin embargo, este algoritmo aplica técnicas de re-planificación ante cambios en el ambiente, como obstáculos o cambios en el terreno [Stentz, 1994].

Algorithm 3 Algoritmo D*: Función PROCESS-STATE

```

1: function PROCESS-STATE
2:    $X \leftarrow \text{MIN-STATE}()$            ▷ Selecciona el estado con el valor mínimo en la cola de
      prioridades
3:   if  $X = \text{NULL}$  then return  $-1$            ▷ Si no hay más estados, finaliza el proceso
4:   end if
5:    $k_{viejo} \leftarrow \text{GET-KMIN}(); \text{DELETE}(X)$    ▷ Obtiene el valor mínimo de la función k y
      elimina el estado  $X$  de la cola
6:   if  $k_{viejo} < h(X)$  then
7:     for cada vecino  $Y$  de  $X$  do           ▷ Revisa todos los vecinos de  $X$ 
8:       if  $h(Y) \leq k_{viejo}$  y  $h(X) > h(Y) + c(Y, X)$  then
9:          $b(X) \leftarrow Y; h(X) \leftarrow h(Y) + c(Y, X)$  ▷ Actualiza la heurística y el mejor
      predecesor
10:      end if
11:    end for
12:  end if
13:  if  $k_{viejo} = h(X)$  then
14:    for cada vecino  $Y$  de  $X$  do
15:      if  $t(Y) = \text{NUEVO}$  o  $(b(Y) = X \text{ y } h(Y) \neq h(X) + c(X, Y))$  o  $(b(Y) \neq X \text{ y } h(Y) > h(X) + c(X, Y))$  then
16:         $b(Y) \leftarrow X; \text{INSERT}(Y, h(X) + c(X, Y))$    ▷ Inserta o actualiza  $Y$  en la
      cola con el nuevo valor de  $h$ 
17:      end if
18:    end for
19:  else
20:    for cada vecino  $Y$  de  $X$  do
21:      if  $t(Y) = \text{NUEVO}$  o  $(b(Y) = X \text{ y } h(Y) \neq h(X) + c(X, Y))$  then
22:         $b(Y) \leftarrow X; \text{INSERT}(Y, h(X) + c(X, Y))$    ▷ Inserta o actualiza  $Y$  en la
      cola si es un nuevo estado o necesita actualización
23:      else
24:        if  $b(Y) \neq X$  y  $h(Y) > h(X) + c(X, Y)$  then
25:           $\text{INSERT}(X, h(X))$  ▷ Reinserta  $X$  en la cola si un vecino tiene un valor
      mejorado
26:        else
27:          if  $b(Y) \neq X$  y  $h(X) > h(Y) + c(Y, X)$  y  $t(Y) = \text{CERRADO}$  y  $h(Y) > k_{viejo}$  then
28:             $\text{INSERT}(Y, h(Y))$            ▷ Actualiza  $Y$  si un mejor camino ha sido
      encontrado desde un estado cerrado
29:          end if
30:        end if
31:      end if
32:    end for
33:  end if return  $\text{GET-KMIN}()$            ▷ Devuelve el siguiente valor mínimo de la función k
34: end function

```

Algorithm 4 Algoritmo D*: Función MODIFY-COST

```

1: function MODIFY-COST( $X, Y, cval$ )
2:    $c(X, Y) \leftarrow cval$                                 ▷ Modifica el costo de la arista entre  $X$  y  $Y$ 
3:   if  $t(X) = \text{CERRADO}$  then
4:     INSERT( $X, h(X)$ )                                ▷ Reinserta  $X$  en la cola si está cerrado y necesita
      actualización
5:   end if return GET-KMIN()                            ▷ Devuelve el siguiente valor mínimo de la función  $k$ 
6: end function

```

2.8.8. RTA* (Real-Time A*)

El algoritmo RTA* es un algoritmo de búsqueda heurística en tiempo real en donde se debe actuar de forma rápida en un ambiente inicial parcialmente conocido. Este algoritmo se caracteriza por realizar una búsqueda local limitada en cada intento y ser capaz de adaptarse a los cambios del ambiente, en donde a través de múltiples iteraciones puede converger a un resultados óptimos [Bulitko y Lee, 2006].

Algorithm 5 Algoritmo RTA* (Real-Time A*)

```

1: function RTA*
2:   Inicializar  $h \leftarrow h_0$                             ▷ Inicializa la función heurística  $h$  con su valor inicial
3:    $s \leftarrow s_{\text{inicial}}$                                 ▷ Asigna el estado inicial  $s$ 
4:   while  $s \notin S_{\text{objetivo}}$  do                            ▷ Continúa hasta que se alcance un estado objetivo
5:     Generar  $S(s, 1) = \{s' \in S \mid \|s, s'\| = 1\}$         ▷ Genera los estados sucesores de  $s$ 
6:      $s' \leftarrow_{s' \in S(s, 1)} (g(s, s') + h(s'))$         ▷ Selecciona el sucesor que minimiza la suma del
      costo y la heurística
7:     if  $g(s, s') + h(s') > h(s)$  then
8:        $h(s) \leftarrow g(s, s') + h(s')$                 ▷ Actualiza la heurística de  $s$  si se encuentra un
      mejor camino
9:     end if
10:     $s \leftarrow s'$                                        ▷ Avanza al siguiente estado
11:  end while
12: end function

```

2.8.9. LRTA* (Learning Real-Time A*)

Es un algoritmo de cálculo de rutas en tiempo real basado en RTA* que, a diferencia de este, ajusta su función heurística de manera dinámica durante la exploración del espacio de búsqueda, permitiendo que el algoritmo aprenda de sus errores y mejore su toma de decisiones a medida que se mueve por un entorno desconocido [Korf, 1990].

Algorithm 6 Algoritmo LRTA*

```

1: function LRTA*( $s_0, goal, h$ )           ▷  $s_0$ : Estado inicial,  $goal$ : Estado objetivo,  $h$ : Función
    heurística
2:    $s \leftarrow s_0$                        ▷ Inicializa el estado actual como el estado inicial
3:   while  $s \neq goal$  do                 ▷ Continúa hasta alcanzar el estado objetivo
4:      $a \leftarrow$  Acción que minimiza  $h(s')$  para cada sucesor  $s'$  de  $s$  ▷ Selecciona la acción
    óptima
5:      $s' \leftarrow$  Resultado de aplicar  $a$  a  $s$  ▷ Calcula el nuevo estado resultante de la acción
6:     if  $s' \neq goal$  then
7:        $h(s) \leftarrow \min_{s''} \{c(s, s'') + h(s'')\}$  ▷ Actualiza la heurística basada en el costo
    y la heurística del sucesor
8:     end if
9:      $s \leftarrow s'$                      ▷ Avanza al siguiente estado
10:  end while
11:  return SUCCESS                       ▷ Devuelve éxito al alcanzar el objetivo
12: end function

```

2.8.10. Comparación algoritmos

Si bien los tres algoritmos mencionados trabajan de manera similar, estos tienen diferencias claves que definen tanto su funcionamiento, su eficiencia y su adaptabilidad a diferentes entornos o circunstancias.

Característica	D*	RTA*	LRTA*
Adaptabilidad a cambios en el entorno	Alta, con replanificación constante ante cambios	Media, con reacciones rápidas pero limitada optimización	Alta, con aprendizaje y ajuste continuo de heurísticas
Eficiencia en la búsqueda	Alta, generalmente encuentra rutas óptimas	Baja, prioriza la rapidez sobre la optimización	Media, mejora con el tiempo pero puede requerir más iteraciones
Uso de recursos	Alto, debido a la necesidad de replanificación y actualización constante	Bajo, con cálculos locales simples	Medio, con almacenamiento de heurísticas aprendidas

Tabla 3: Comparación entre los algoritmos D*, RTA* y LRTA*

2.9. Optimización de rutas

La optimización de rutas no solo se enfoca en encontrar el camino más eficiente entre dos puntos, sino también en simplificar la representación de estas rutas para facilitar su comprensión y utilización. En el contexto de aplicaciones de navegación en interiores, la ruta

óptima generada puede contener una gran cantidad de puntos debido a la complejidad del entorno. Para abordar este problema, se emplean técnicas o algoritmos de simplificación de curvas.

2.9.1. Algoritmo de Ramer-Douglas-Peucker

En el contexto de esta memoria, una ruta se compone de una serie de nodos que representan una posición en el plano. El algoritmo de Ramer-Douglas-Peucker es un algoritmo utilizado para reducir el número de nodos que componen una curva (o ruta), minimizando la longitud o la complejidad de esta. En el contexto de la generación de rutas para aplicaciones de navegación, es importante que las rutas sean lo más simples y claras posible, de tal manera que el usuario final no tenga problemas para llegar a su objetivo [Pinandito *et al.*, 2023].

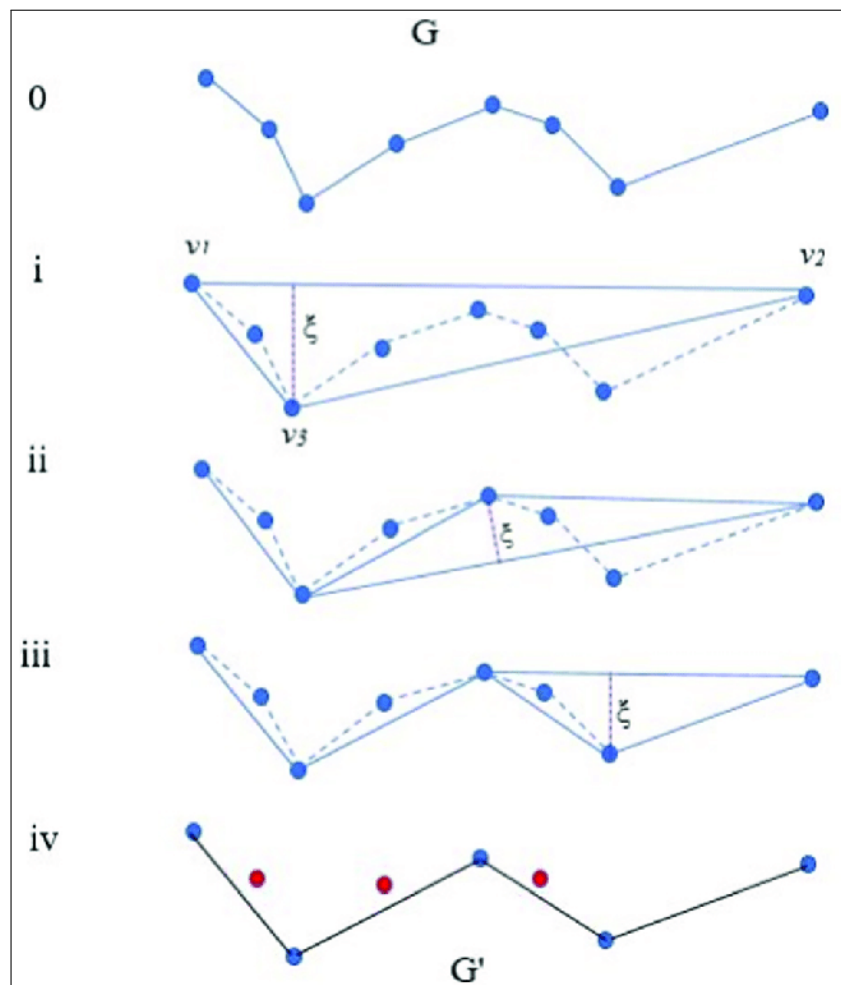


Figura 8: Representación proceso de algoritmo de Ramer-Douglas-Peucker.
Fuente: [Farjad *et al.*, 2019]

2.9.2. Algoritmo de Visvalingam-Whyatt

El algoritmo de Visvalingam-Whyatt es un algoritmo enfocado en la simplificación de rutas. Este metodo utilizá el área de los triángulos formados por puntos consecutivos en la ruta, a diferencia del algoritmo de Ramer-Douglas-Peucker, que se basa en una distancia máxima pre definida [Visvalingam y Whyatt, 1993].

El algoritmo funciona eliminando el punto que forma el triángulo con el área más pequeña entre sus puntos consecutivos de manera iterativa. Este proceso se repite hasta alcanzar un umbral de simplificación.

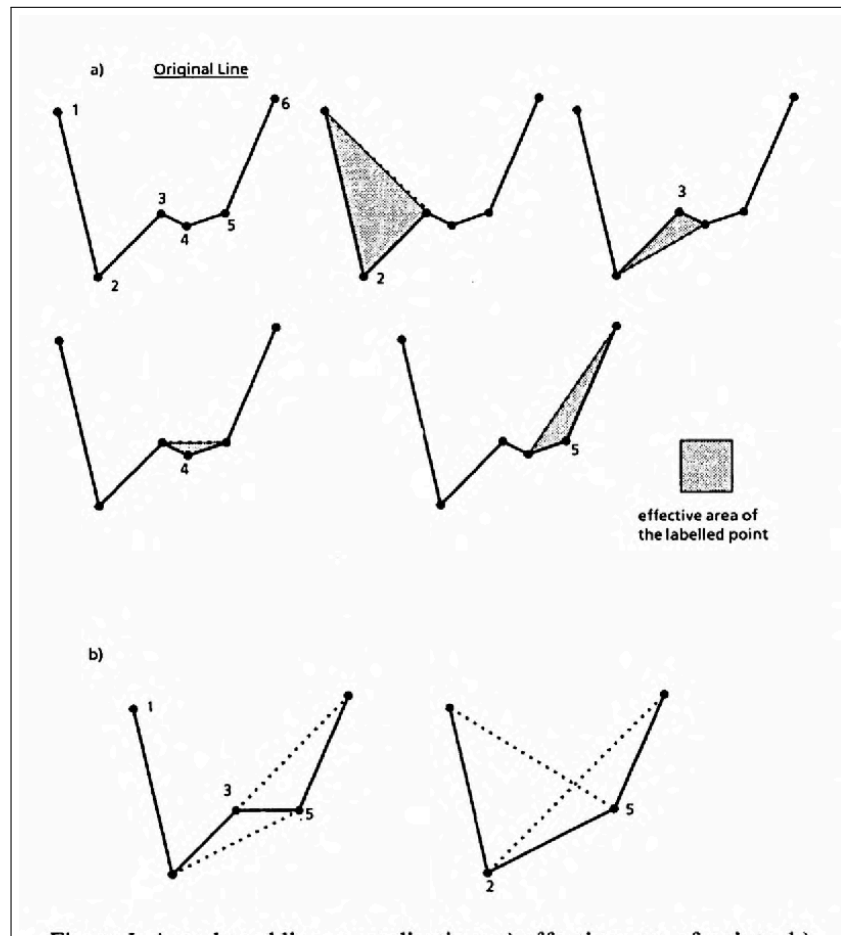


Figura 9: Representación algoritmo de Visvalingam-Whyatt.
Fuente: [Visvalingam y Whyatt, 1993]

2.9.3. Algoritmo de Lang

El algoritmo de Lang es una modificación del algoritmo de Ramer–Douglas–Peucker que, a diferencia de este, utiliza una banda de tolerancia alrededor de la línea original para determinar qué puntos mantener. El proceso implica crear una banda de tolerancia alrededor de la línea original, eliminar los puntos que caen dentro de esta banda, y luego ajustar la banda basándose en los puntos restantes, repitiendo el proceso hasta que no pueda eliminar mas puntos [Lang, 1969].

2.10. Imperfecciones en rutas generadas por algoritmos de generación de rutas

En los algoritmos de búsqueda de rutas es común presentar ciertas imperfecciones que afectan a la calidad de la solución. Entre los mas comunes se encuentran:

2.10.1. Zigzag y “Dientes de sierra”

Los zigzag se caracterizan por cambios frecuentes en la dirección de la ruta, mientras que los “dientes de sierra” son pequeñas desviaciones repetitivas que crean un patrón similar al borde de una sierra. Ambos ocurren cuando el algoritmo intenta minimizar la distancia total en un espacio discreto, resultando en patrones poco naturales y difíciles de seguir para un usuario [Ferguson *et al.*, 2005].

2.10.2. Oscilaciones y adherencia a los bordes

Las oscilaciones ocurren cuando la ruta oscila innecesariamente alrededor de una línea recta ideal. Por otro lado, la adherencia a los bordes se produce cuando las rutas se pegan excesivamente a los obstáculos. Ambos problemas están relacionados con la optimización de la distancia en un espacio discreto y pueden resultar en rutas que, aunque técnicamente óptimas, son poco prácticas o incómodas para los usuarios [Choset *et al.*, 2005].

2.10.3. Cambios bruscos de dirección y falta de suavizado

Al aplicar algoritmos de búsqueda en espacios discretos, como cuadrículas, las rutas pueden presentar giros de 90 grados o más. Aunque estos giros son matemáticamente correctos para minimizar la distancia, resultan poco naturales para el movimiento humano, provocando que

las rutas parezcan artificiales o robóticas, lo que puede dificultar su seguimiento en la práctica [Latombe, 1991].

2.11. Procesamiento de Imágenes

El procesamiento de imágenes es fundamental para el análisis de planos, permitiendo extraer información valiosa del espacio a utilizar, como zonas transitables, obstáculos y puntos de interés.

2.11.1. Técnicas de Procesamiento de Imágenes

El procesamiento de imágenes permite transformar planos digitales en representaciones computacionales para su posterior análisis.

Segmentación de Imágenes La segmentación es un proceso que divide una imagen en zonas de interés, útil para identificar áreas transitables, obstáculos y otros elementos relevantes en un plano. Las principales técnicas incluyen:

- **Umbralización:** Separa objetos del fondo basándose en la intensidad de los píxeles. Es útil para diferenciar entre áreas transitables y no transitables en planos.
- **Detección de bordes:** Identifica límites entre diferentes áreas en un plano. Esta técnica se utiliza para reconocer las zonas de un espacio, como tiendas, pasillos, etc.
- **Crecimiento de regiones:** Agrupa píxeles similares para identificar áreas específicas. Es eficaz para reconocer zonas continuas como salas o áreas abiertas en un plano.

Estas técnicas, permiten una interpretación profunda de los planos, facilitando la creación de representaciones digitales [Szeliski, 2010].

Detección de Objetos y Puntos de Interés Además de la segmentación, la detección de objetos específicos y puntos de interés es importante para el análisis de planos. Esto incluye:

- **Detección de contornos:** Útil para identificar formas específicas como escaleras, ascensores o señalizaciones en el plano.
- **Reconocimiento de patrones:** Permite identificar símbolos o iconos que representan puntos de interés como tiendas, oficinas o servicios.

Estas técnicas ayudan a extraer información útil de los planos a utilizar [Gonzalez y Woods, 2018].

2.11.2. Tecnologías y Herramientas

Para el procesamiento de imágenes se utilizan diversas tecnologías y bibliotecas especializadas, entre las que se encuentran:

OpenCV OpenCV es una biblioteca de código abierto ampliamente utilizada en visión por computador y procesamiento de imágenes. Tiene integrado una gran cantidad de algoritmos para tareas como detección de objetos, reconocimiento facial, seguimiento de movimiento, y manipulación de imágenes. Es una herramienta fundamental en campos como la robótica, el reconocimiento de patrones y la realidad aumentada [Bradski y Kaehler, 2008].

NumPy NumPy es una biblioteca utilizada en la computación científica en Python, especializada en el manejo eficiente de arreglos multidimensionales. Contiene una gran cantidad de herramientas para realizar operaciones matemáticas complejas, transformaciones de datos y análisis estadísticos. [Oliphant, 2006].

2.12. Aplicación Móvil

Dado uno de los objetivos de la memoria de realizar una aplicación para ubicarse y obtener información relevante dentro de un espacio cerrado y concurrido, es deseable que la aplicación sea capaz de ejecutarse en la mayor cantidad de dispositivos móviles posibles. Por lo tanto, para lograr este objetivo se eligió entre desarrollar una aplicación nativa o una aplicación híbrida, las cuales se definen como:

- **Aplicación Nativa:** Es una aplicación que se desarrolla para un sistema operativo en específico, como iOS o Android, y que usualmente utilizan un lenguaje de programación nativo, como Swift o Kotlin respectivamente.
- **Aplicación Híbrida:** Es una aplicación que se desarrolla usando frameworks de terceros, como React Native o Flutter, y que permiten exportar una aplicación a distintos sistemas operativos utilizando el mismo código fuente.

Dado que durante el 2023, el sistema operativo iOS de Apple correspondía al 28 % de la cuota mundial y Android de Google el 71 % [Statista, 2024], se decidió realizar una aplicación híbrida que pueda ser exportada a estos dos sistemas operativos. Para esto, se seleccionó

React Native como el framework para desarrollar la aplicación móvil. React Native es un framework de código abierto desarrollado y mantenido por Meta Platforms, el cual permite desarrollar aplicaciones multiplataforma utilizando TypeScript y una estructura similar a la utilizada en el desarrollo web. También se seleccionó Expo para facilitar el desarrollo, el cual es un framework de código abierto para aplicaciones desarrolladas en React Native. Este framework encapsula una serie de configuraciones básicas y paquetes útiles para agilizar el proceso de desarrollo, con el fin de enfocarnos en la construcción de la aplicación como tal.

2.13. Estado del Arte

El desarrollo de sistemas de posicionamiento en interiores ha cobrado una importancia significativa en la última década debido a la creciente demanda de aplicaciones que requieren ubicaciones precisas dentro de edificios y estructuras complejas. Mientras que el Sistema de Posicionamiento Global (GPS) ha revolucionado la navegación y la localización en exteriores, sus limitaciones en entornos cerrados han impulsado la investigación y desarrollo de tecnologías alternativas para interiores [Gu *et al.*, 2009].

Este Estado del Arte se centra en revisar las tecnologías actuales y los métodos utilizados para el posicionamiento en interiores, así como los algoritmos de generación y optimización de rutas. Se explorarán los sistemas basados en Wi-Fi, Bluetooth y otras tecnologías emergentes, evaluando sus ventajas, limitaciones y casos de uso.

2.13.1. Comparación de Tecnologías de Posicionamiento en Interiores

Existen múltiples tecnologías y métodos para obtener la ubicación en espacios cerrados internos, las cuales tienen distintos costos de implementación, escalabilidad o precisión. En el trabajo *A Survey of Indoor Location Technologies, Techniques and Applications in Industry* Hayward *et al* [Hayward *et al.*, 2022], los autores presentan una comparación de las tecnologías más utilizadas para la localización interna.

Tecnología	Costo (Receptor)	Precisión	Cobertura	Consumo de Energía	Línea de Vista	Limitación Principal
GNSS	£10-£50	3-15 m+	Global	>150 mW-1.5 W	Sí	Multipath e interferencias
Wi-Fi	£30-£150+	0.4-5 m	10-2500 m ²	>500 mW	No	Interferencia ambiental
Bluetooth LE	£5-£30	0.9-2 m	10-1000 m ²	1-50 mW	No	Baja precisión
UWB	£10-£200	0.3-0.5 cm	10-400 m ²	>1 W	No	Alto consumo de energía
Acústico	£10-£20	0.1-0.6 m	10-2500 m ²	>300 mW	Sí	Interferencia de sonido ambiente
Visión	£10-£250	1 mm-0.3 m	~6 m	>1 W	Sí	Requiere condiciones de iluminación
UHF RFID (Pasivo)	£0.01-£0.03 +£50-£100	1-5 m	10-1000 m ²	>300 mW	No	Impacto severo de metal Alto costo de antena y lector
UHF RFID (Activo)	£50-£100+ +£50-£100	0.1-1 m	10-1000 m ²	>250 mW (Lector)	No	Interferencia de metal Alto costo unitario

Tabla 4: Resumen de enfoques tecnológicos y precisiones correspondientes para la investigación y tecnologías IPS disponibles comercialmente.

Fuente: [Hayward *et al.*, 2022]

En esta tabla podemos ver una comparación de varias tecnologías aplicadas a la detección de dispositivos en espacios interiores. Se puede ver la información técnica de tecnologías como redes Wi-Fi y Bluetooth, las cuales son mencionadas en esta memoria. De las redes Wi-Fi, podemos ver que tienen un costo de implementación relativamente alto, sin embargo, es bastante común que los espacios objetivo de este desarrollo (centros comerciales, universidades o oficinas), ya tengan una red Wi-Fi implementada previamente. Además, características como su alta escalabilidad o precisión de entre 0.4[m] y 5[m], hacen de Wi-Fi una opción atractiva para soluciones de localización en interiores. En contraste, el Bluetooth, especialmente en su variante Bluetooth Low Energy (BLE), si bien presenta un menor costo, esta tecnología tiene una baja cobertura en comparación al Wi-Fi. Sin embargo, dado que existen puntos de acceso Wi-Fi con Bluetooth Low Energy integrado, la utilización de esta tecnología en paralelo al Wi-Fi podría mejorar la precisión de los resultados finales.

Por otro lado, este trabajo presenta otras tecnologías menos utilizadas como la identificación por radiofrecuencias (RFID), Ultra Wideband o utilizando cámaras (visión). Si bien estas alternativas son validas e incluso ofrecen una mayor precisión en comparación al Wi-Fi o Bluetooth, utilizarlas en un ambiente real significaría una inversión alta en comparación a la antes mencionadas. Además, el procesamiento de los datos para obtener la posición de los objetos es más complicado, como por ejemplo en la utilización de cámaras, ya que estas deben ser capaces de diferenciar personas y calcular su posición desde su perspectiva.

2.13.2. Google Maps Indoor Location

Google Maps es un servicio gratuito de mapas y navegación proporcionado por la empresa estadounidense Alphabet. Ofrece una serie de servicios entre los cuales se encuentra: información de servicios y puntos de interés, fotografías satelitales, generación de rutas, información en tiempo real entre otras. Durante el año 2011, la versión 6.0 de Google Maps estrenó la funcionalidad de mostrar información interna de edificios [Isaac, 2011]. Específicamente, la aplicación móvil permite visualizar puntos de interés en los diferentes pisos de un espacio cerrado, como lo es un estadio, una estación de tren o un centro comercial [Google Maps, 2024a].

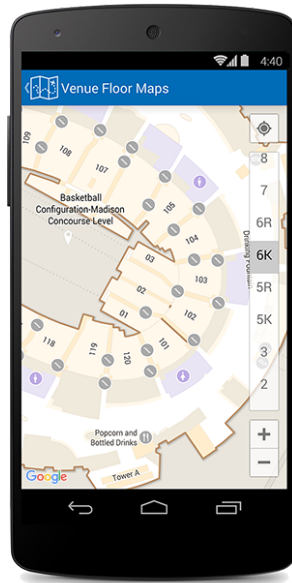


Figura 10: Interfaz de Google Maps en un dispositivo móvil.
Fuente: [Google Maps, 2024a]

Sin embargo, desde su lanzamiento en 2011, dicha funcionalidad no ha sido utilizada de forma masiva, especialmente en Chile, donde a día de hoy (Mayo 2024), ningún punto de interés de la capital esta integrada a esta funcionalidad, entre los que se encuentran: El Estadio Nacional Julio Martínez Prádanos, el centro comercial Cenco Costanera, uno de los más grandes de Latinoamérica o el Parque Metropolitano de Santiago. Además, Google Maps no cuenta con una integración publica para obtener la ubicación en tiempo real dentro de estos espacios, limitándose a funcionar como un sistema de mapas estático que no interactúa activamente con la localización dinámica de los usuarios dentro de los espacios cerrados.

2.13.3. Apple Maps Indoor Positioning

Por otro lado, la aplicación nativa de mapas y navegación, Apple Maps, también ofrece una solución de localización interna. Esta funcionalidad fue lanzada en la Conferencia Mundial de Desarrolladores de Apple (WWDC) del año 2019 [Inc., 2019]. En ese momento, se lanzó el programa de mapeo de interiores para organizaciones publicas o privadas con un gran espacio con el fin de entregar información del entorno a sus usuarios. Además, a diferencia de Google Maps, Apple Maps permite la integración con redes Wi-Fi ya existentes para obtener la ubicación en tiempo real dentro de estos espacios con un nivel de precisión similar al GPS. A pesar de contar con estas funcionalidades, Apple solo permite que aeropuertos puedan mostrar su información publica en la aplicación nativa, sin embargo, permite que otros espacios como hospitales, hoteles, museos, entre otros, utilicen esta integración en aplicaciones propias utilizando dispositivos iOS [Inc., 2024].

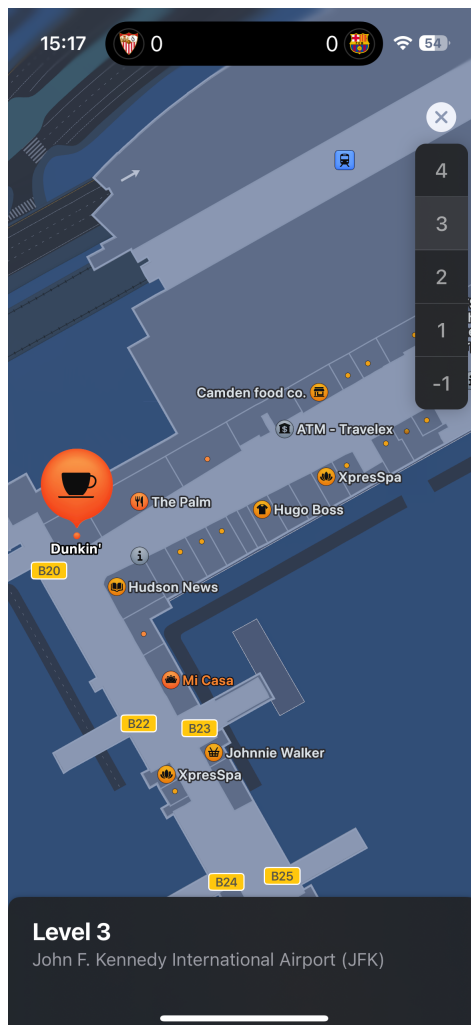


Figura 11: Interfaz de Apple Maps en un dispositivo móvil.
Fuente: Elaboración propia

Por otro lado, al igual que Google Maps, la integración de puntos de interés en la capital de Chile, tales como el aeropuerto de Santiago o centros comerciales como Cenco Costanera es nula², limitándose a estar disponible en los principales aeropuertos del mundo, o en puntos de interés dentro de los Estados Unidos de América.

²Revisado en julio de 2024.

2.13.4. Sistema IndoorAtlas

IndoorAtlas es una innovadora solución de posicionamiento en interiores que utiliza una combinación de tecnologías, con énfasis en el campo magnético terrestre. Desarrollada por una empresa finlandesa, esta tecnología aprovecha las variaciones naturales del campo magnético dentro de las estructuras de los edificios para crear "huellas magnéticas" únicas de los espacios interiores [ind, 2024].

El sistema funciona de la siguiente manera:

1. Mapeo: Se crea un mapa magnético del edificio utilizando los sensores de los smartphones.
2. Posicionamiento: Los dispositivos de los usuarios detectan el campo magnético local y lo comparan con el mapa previamente creado.
3. Fusión de datos: IndoorAtlas combina esta información magnética con otras señales como WiFi, Bluetooth y datos de los sensores inerciales para mejorar la precisión.

Una de las principales ventajas de IndoorAtlas es que no requiere hardware adicional más allá de los smartphones de los usuarios, lo que lo hace altamente escalable y económico de implementar. Según la empresa, el sistema puede alcanzar una precisión de 1 a 3 metros en condiciones óptimas [IndoorAtlas, 2023].

2.13.5. Proyecto IPIN (Indoor Positioning and Indoor Navigation)

El proyecto IPIN (Indoor Positioning and Indoor Navigation) es una iniciativa internacional que busca avanzar en el campo del posicionamiento y la navegación en interiores. Una de sus contribuciones más significativas es la organización de competencias anuales que evalúan diferentes tecnologías y algoritmos de posicionamiento en interiores en condiciones del mundo real [ipi, 2024].

Las competencias IPIN se caracterizan por:

- Escenarios realistas: Los participantes deben demostrar sus soluciones en edificios reales con diversas condiciones ambientales.
- Múltiples tracks: La competición incluye diferentes categorías, como soluciones basadas en smartphones, sistemas de posicionamiento de peatones, y soluciones para robots móviles.
- Fomento de la innovación: Al reunir a investigadores y profesionales de todo el mundo, IPIN promueve el intercambio de ideas y el avance del campo.

Los resultados de estas competencias proporcionan una visión valiosa del estado actual de las tecnologías de posicionamiento en interiores y ayudan a identificar las soluciones más prometedoras. Por ejemplo, en recientes ediciones, se ha observado un aumento en la precisión de las soluciones basadas en smartphones, con algunos sistemas alcanzando errores medios por debajo de los 2 metros [Potorti *et al.*, 2023].

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

La propuesta de solución de esta memoria consiste en el desarrollo de un sistema de navegación en interiores basado en redes Wi-Fi. El sistema se compone de una aplicación móvil híbrida desarrollada en React Native, que se comunica con un servidor backend en Node.js. Para la localización, se utiliza la infraestructura de red Wi-Fi de Cisco Meraki, la cual es capaz de detectar la posición de los dispositivos utilizando sus puntos de acceso, y la información recibida es procesada mediante un filtro de Kalman para mejorar la precisión. La generación de rutas se realiza utilizando el algoritmo A* con una heurística de distancia de Chebyshev, y la ruta obtenida es simplificada posteriormente utilizando el algoritmo de Ramer-Douglas-Peucker. Finalmente, para visualizar la información se procesó un plano del espacio de pruebas, obteniendo así una representación 3D en donde se puede visualizar las rutas, posición actual e información de las tiendas.

3.1. Metodología utilizada

Para el desarrollo de esta solución, se utilizó una metodología que combina la teoría de grafos y algoritmos de búsqueda de rutas, integrados en una aplicación móvil para proporcionar una aproximación a la navegación en interiores basada en redes Wi-Fi. A continuación, se describen los pasos y procesos involucrados en esta metodología.

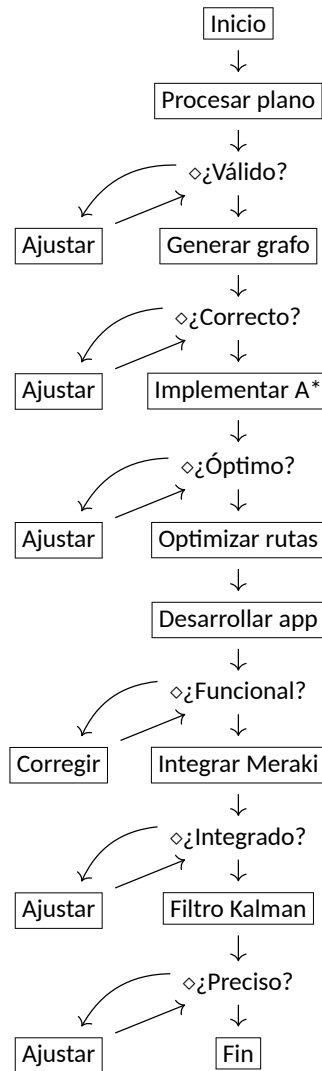


Figura 12: Diagrama de flujo compacto del proceso de desarrollo

3.2. Generación de Mapa

Inicialmente, se comenzó considerando cómo obtener la información del espacio físico que se utilizaría como espacio de pruebas. Para lograr este objetivo, el principal enfoque era no utilizar hardware de terceros para escanear los espacios físicos. Además, para el enfoque de esta memoria no se requiere un alto nivel de detalle ni fotografías del espacio, sino que se necesita una representación simple en donde se detalle la división de los espacios. Por lo tanto, como información de entrada se utilizó un plano del espacio de pruebas, específicamente un plano técnico de la construcción de un centro comercial. El plano [13] describe todos los puntos de interés, tales como las tiendas, entradas, estacionamientos, etcétera.

Aunque la visualización final es en un plano 3D para mejorar la experiencia del usuario y visualización del ambiente, el procesamiento del mapa, los cálculos de rutas y posición se realizan en un plano 2D, ya que los datos obtenidos solo tienen 2 dimensiones, latitud y longitud.. Esta aproximación permite aprovechar las ventajas de los algoritmos de procesamiento 2D mientras se ofrece una representación visual más atractiva y comprensible en 3D.

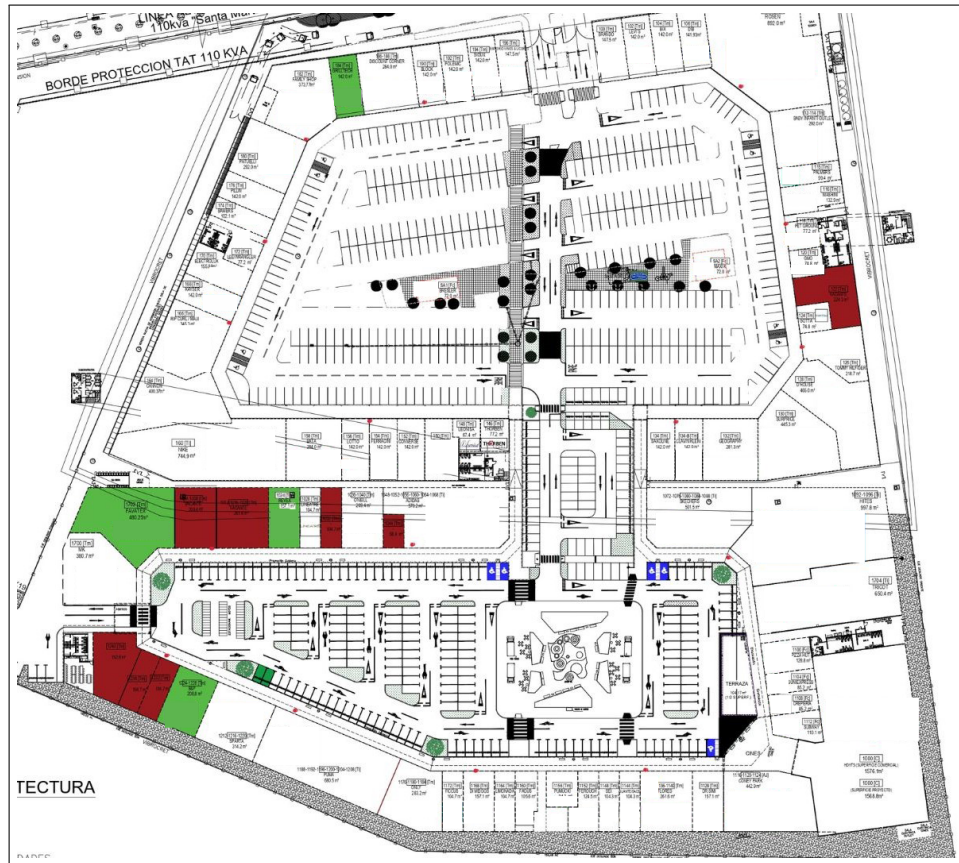


Figura 13: Plano original espacio de prueba.
Fuente: Elaboración propia.

Una vez obtenido el plano del espacio de ejemplo, con sus tiendas, espacios transitables y estacionamientos definidos, fue necesario poder identificar estos espacios para el procesamiento del plano. Si bien se analizó la opción de utilizar algún método automático, como inteligencia artificial, para identificar estos espacios de forma automática, se optó por hacerlo manualmente, ya que ese no es el enfoque principal de la memoria, además que el identificar los espacios fue un trabajo de una vez y que no tomó demasiado tiempo, completándose en aproximadamente una hora.



Figura 14: Mapa procesado manualmente, en donde el rojo indica las tiendas, amarillo las zonas transitables, cian los estacionamientos, negro las paredes y blanco las zonas no identificadas.

Fuente: Elaboración propia.

Una vez generado el plano procesado con los espacios identificados, se seleccionaron una serie de coordenadas geográficas en conjunto a su posición correspondiente en píxeles dentro del plano. Estas coordenadas geográficas (latitud y longitud) fueron convertidas a su representación en píxeles en el plano digital utilizando un método de regresión lineal multivariable, permitiendo así una referencia física dentro del entorno virtual. Este método crea una transformación lineal entre las coordenadas geográficas y de píxeles, utilizando el algoritmo de mínimos cuadrados para obtener los coeficientes óptimos. Esta conversión es crucial para la implementación de la navegación en interiores, ya que permite relacionar la posición real de un dispositivo con su ubicación en el plano, facilitando la generación de rutas y la visualización de la ubicación en tiempo real.

Luego, se generó un grafo para representar las zonas transitables del plano, las cuales corresponden a las zonas del centro comercial en donde los clientes pueden transitar, tales como pasillos o tiendas. Para esto, se dividió la imagen en cuadrados de píxeles que repre-

sentan $4 [m^2]$, lo cual se calculó utilizando los puntos de referencia previamente explicados. El tamaño de las cuadrículas se definió analizando la información entregada por Cisco Mera-ki, en donde las detecciones tienen un error asociado de al menos 2 metros, por lo que un tamaño menor no tendría sentido. Una vez dividido, se iteró sobre cada cuadrícula aplicando la siguiente lógica: Si el centro del cuadrado se encuentra en una zona transitable, esta se agrega al conjunto de nodos transitables. En caso contrario, se ignora. Una vez obtenido el conjunto de nodos transitables, se itera sobre cada nodo revisando sus nodos vecinos y uniendo los nodos transitables contiguos, con el fin de generar un grafo de espacios transitables.

Después, se identificó cada tienda obteniendo los contornos de sus áreas (de color rojo). Seguidamente, se iteró sobre cada una obteniendo los centros geométricos de cada área. Posteriormente, se añadió cada centro a la lista de nodos disponibles, y se generó la conexión entre el centro de cada tienda y el nodo válido transitable más cercano, es decir, que no exista un obstáculo entre ambos. Una vez completado este proceso, se obtuvo dos estructuras de datos principales: Un conjunto de nodos transitables con su posición en píxeles, y un grafo que representa los nodos y sus aristas. Finalmente, se generó una matriz de adyacencia para facilitar la generación de rutas que se implementará posteriormente.

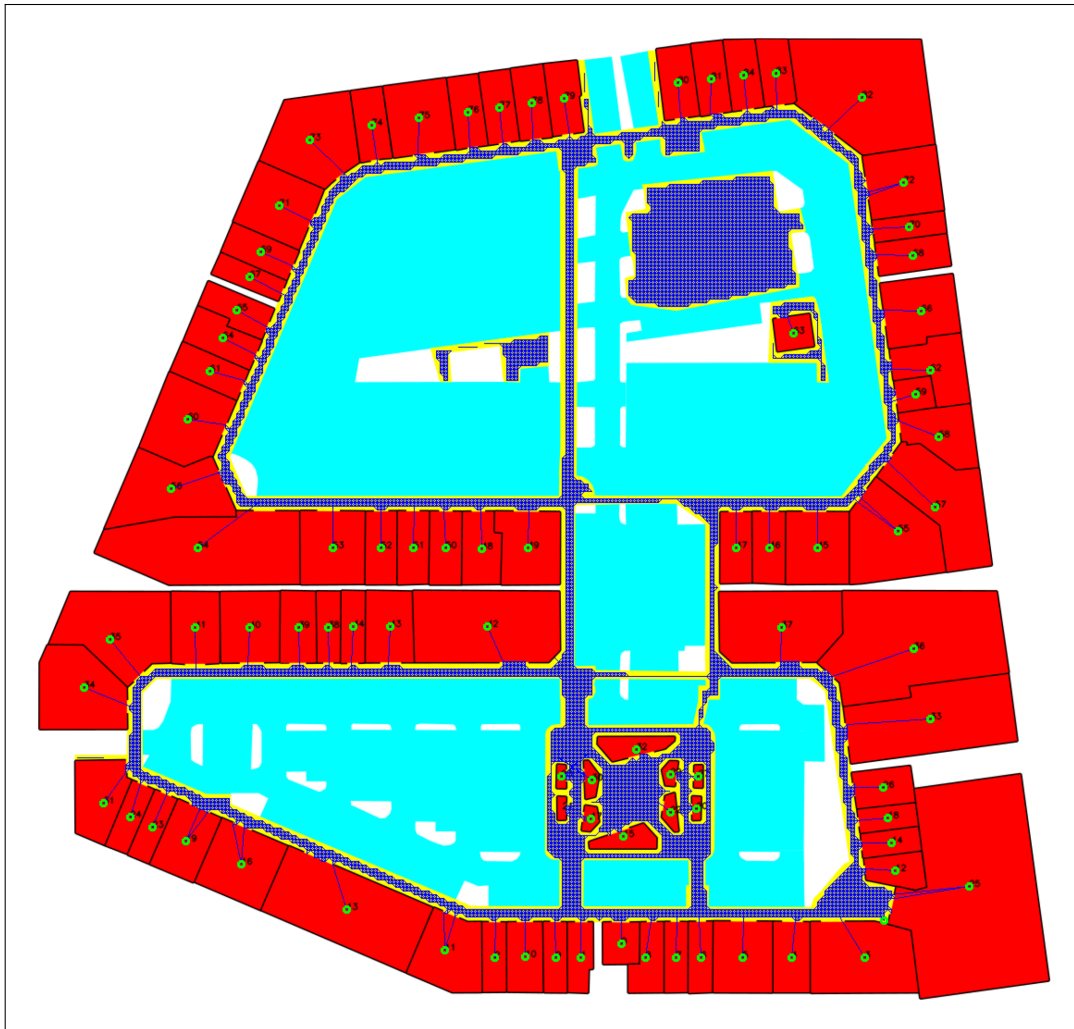


Figura 15: Mapa procesado en donde las zonas azules corresponden a las zonas transitables y los puntos verdes a el centro de cada tienda.

Fuente: Elaboración propia.

3.3. Generación de Rutas

Una vez obtenidas las estructuras de datos: el conjunto de nodos transitables con su posición y la matriz de adyacencia, se comenzó a realizar la lógica para obtener las rutas entre dos nodos. Durante este proceso se analizó qué algoritmo se utilizaría para esta tarea. Dentro de los algoritmos de generación de ruta se encuentran los algoritmos online, que son aquellos que se adaptan a escenarios dinámicos o con conocimiento limitado. Por otro lado, existen los algoritmos offline, que son aquellos que tienen conocimiento completo del ambiente al inicio de la ejecución y que no necesitan adaptarse a cambios en el escenario. Dada la naturaleza estática del escenario utilizado en este proyecto, se optó por utilizar algoritmos offline, entre los cuales se encuentran:

Algoritmo de Dijkstra Encuentra la ruta mas corta desde un nodo origen a todos los demás nodos, permitiendo solamente nodos con pesos positivos [Madkour *et al.*, 2017].

Algoritmo A* Utiliza una heurística, dependiendo del objetivo del algoritmo, para encontrar la ruta mas corta entre un nodo de origen y un nodo objetivo.

Una vez planteados estos algoritmos y el grafo que representan los nodos de las zonas transitables, en donde las aristas representan distancias las cuales son siempre positivas, además que en el contexto de esta memoria se necesita la ruta más corta entre un par de nodos, se decidió utilizar el algoritmo A* aplicando una heurística para obtener un camino optimizado al escenario. Esta elección se basó en varias consideraciones:

- El algoritmo A* destaca por su eficiencia en escenarios estáticos como el grafo utilizado. A diferencia de otros algoritmos como RTA*, LRTA* o D*, que están diseñados específicamente para entornos dinámicos o con información incompleta, A* utiliza la naturaleza estática del grafo para ofrecer un rendimiento superior.
- Dada la necesidad del proyecto de encontrar la ruta más corta entre dos nodos, el algoritmo A* resulta ser la mejor opción ya que garantiza una ruta óptima si es que existe. Al contrario de algoritmos online como RTA*, LRTA* o D*, que no pueden asegurar un resultado óptimo.
- A* se destaca por el uso de una función heurística para guiar la búsqueda hasta el nodo objetivo. Esta función permite al algoritmo priorizar los nodos que parecen estar más cerca del final, lo que resulta en una búsqueda más dirigida y eficaz. Esta característica hace que A* sea particularmente adecuado para el contexto de esta memoria.

Una heurística típica es utilizar la distancia euclidiana entre dos punto, es decir, guiar la búsqueda en los nodos que minimizan la distancia hasta el objetivo.

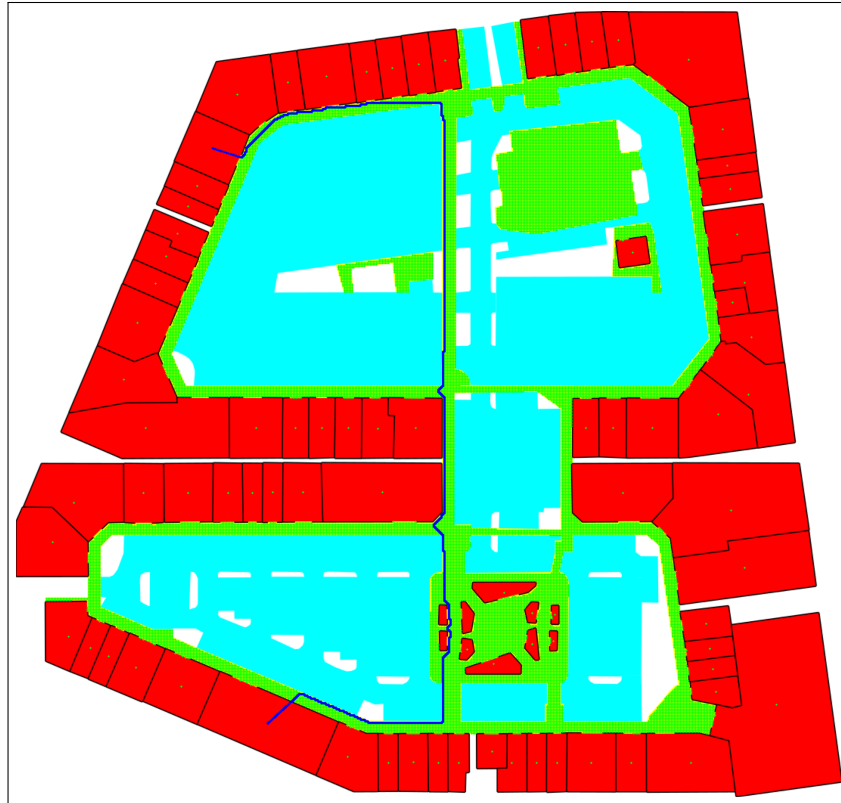


Figura 16: Ruta generada (en azul) utilizando el algoritmo A* con distancia euclidiana como función heurística

Fuente: Elaboración propia.

Si bien la utilización del algoritmo A* en conjunto a una heurística de distancia euclidiana cumple su objetivo de encontrar rutas óptimas en un ambiente real como lo es un centro comercial, las rutas generadas pueden tener ciertas imperfecciones que afectan al objetivo final de esta memoria, como por ejemplo zigzags, “dientes de sierra” o que incluso atraviese obstáculos.

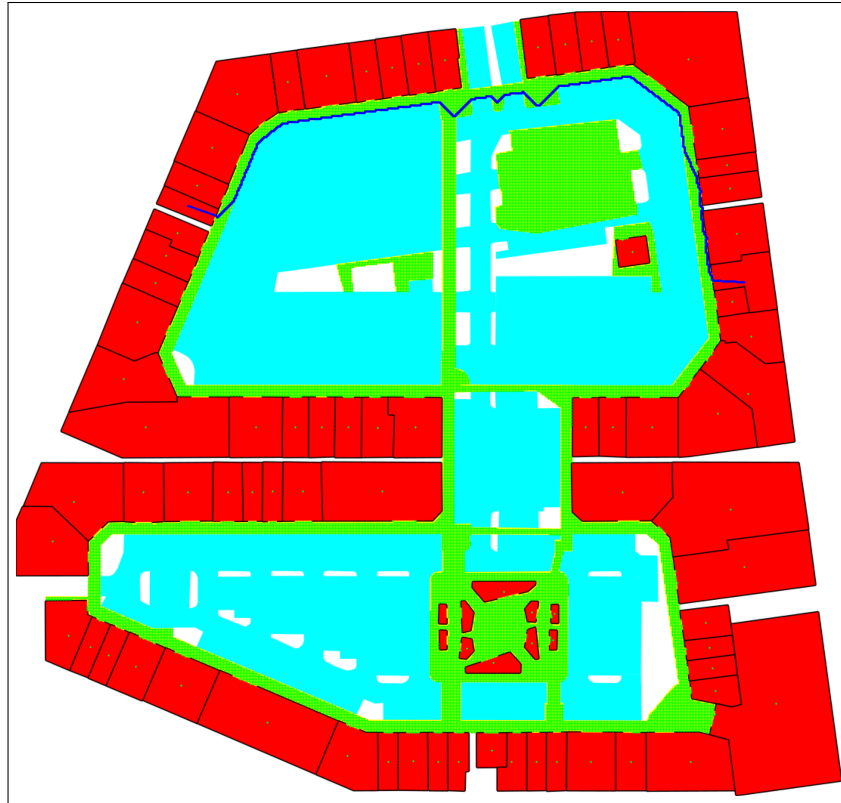


Figura 17: Ruta generada (en azul) utilizando el algoritmo A* con distancia euclidiana como función heurística

Fuente: Elaboración propia.

Dado esta problemática, se aplicó una función heurística distinta en el algoritmo A*, además que se utilizó un algoritmo de aproximación de curvas, con el fin de hacer la ruta menos compleja para el usuario.

3.3.1. Optimización de Rutas

Para optimizar la ruta generada por el algoritmo A*, inicialmente se aplicó una heurística distinta a la euclidiana que genere una ruta mas naturales y mas directas. Para lograr este objetivo se analizaron dos funciones heurísticas distintas, la distancia de Chebyshev y la distancia Manhattan. Estas funciones heurísticas son bastantes similares, ya que son óptimas para movimientos en grillas. La diferencia principal entre ambas funciones es que la distancia Manhattan es más óptima para movimientos ortogonales, mientras que la distancia Chebyshev también permite movimientos diagonales, por lo que es mas compatible dado el grafo modelado.

$$d_{\text{Chebyshev}}(x_1, y_1, x_2, y_2) = \text{máx}(|x_2 - x_1|, |y_2 - y_1|)$$

La distancia de Chebyshev se utiliza principalmente en escenarios cuadriculados que permiten movimientos ortogonales y diagonales, como el plano grafo utilizado. De esta manera, el algoritmo A* se ve influenciado para hacer movimientos ortogonales o diagonales directos. Por otro lado, la distancia euclidiana evalúa la distancia directa entre los nodos evaluados, provocando que, cuando se encuentra con un obstáculo, prefiera rodearlo en lugar de hacer movimientos directos.

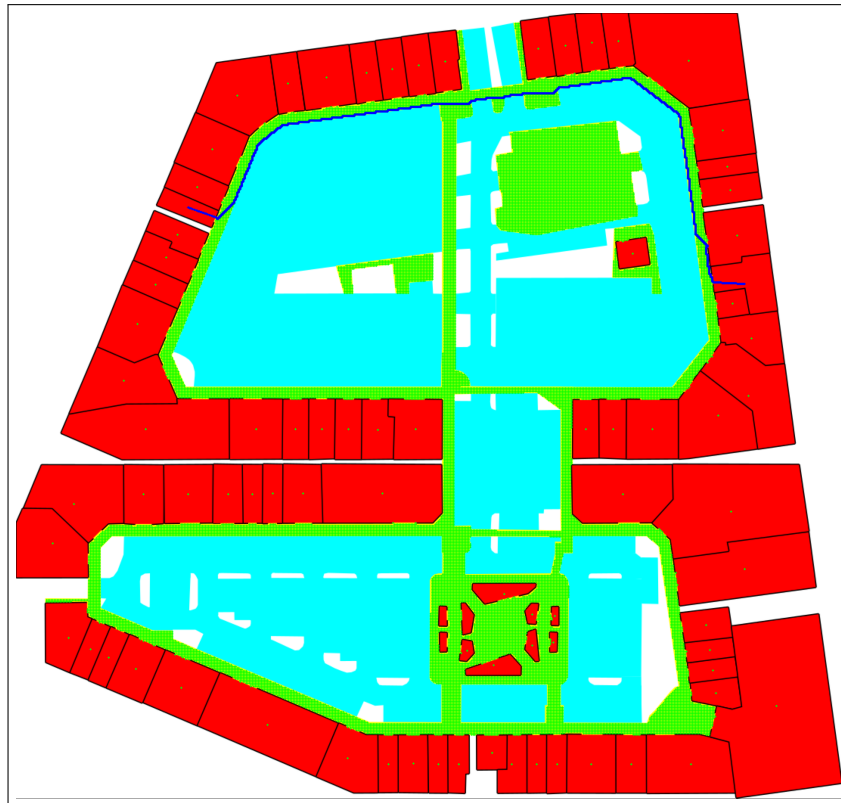


Figura 18: Ruta generada (en azul) utilizando el algoritmo A* con distancia de Chebyshev como función heurística

Fuente: Elaboración propia.

Una vez modificado el algoritmo A* para obtener rutas directas, era necesario simplificar la composición de las rutas. Para esto, se utilizó el algoritmo de Ramer-Douglas-Peucker, el cual se utiliza para disminuir el número de puntos utilizados en la aproximación de una curva, es decir, obtener una aproximación a la ruta generada pero compuesta de menos nodos. Este algoritmo funciona de la siguiente manera:

1. Comienza trazando una recta entre el nodo inicial y el nodo final.

2. Selecciona el nodo mas lejano a la recta anterior.
3. Verifica si la distancia es menor a un ϵ predefinido. Si la distancia es menor, el nodo se elimina. En caso contrario, significa que este nodo es significativo y no se puede eliminar.
4. Se generan dos trazos desde el nodo inicial al previamente seleccionado y desde este hasta el final.
5. Se aplica la lógica de manera recursiva a todos los segmentos generados.

O representado en pseudo-código como

Algorithm 7 Algoritmo de Douglas-Peucker

```
1: function DouglasPeucker(ListaPuntos, epsilon)
2:    $d_{max} \leftarrow 0$ 
3:    $indice \leftarrow 0$ 
4:    $fin \leftarrow longitud(ListaPuntos)$ 
5:   for  $i \leftarrow 2$  to  $fin - 1$  do
6:      $d \leftarrow DistanciaPerpendicular(ListaPuntos[i], Linea(ListaPuntos[1], ListaPuntos[fin]))$ 
7:     if  $d > d_{max}$  then
8:        $indice \leftarrow i$ 
9:        $d_{max} \leftarrow d$ 
10:    end if
11:  end for
12:  if  $d_{max} > epsilon$  then
13:     $resultadosRec1 \leftarrow DouglasPeucker(ListaPuntos[1 : indice], epsilon)$ 
14:     $resultadosRec2 \leftarrow DouglasPeucker(ListaPuntos[indice : fin], epsilon)$ 
15:     $ListaResultado \leftarrow concatenar(resultadosRec1[1 : fin - 1], resultadosRec2)$ 
16:  else
17:     $ListaResultado \leftarrow [ListaPuntos[1], ListaPuntos[fin]]$ 
18:  end if
19:  return  $ListaResultado$ 
20: end function
```

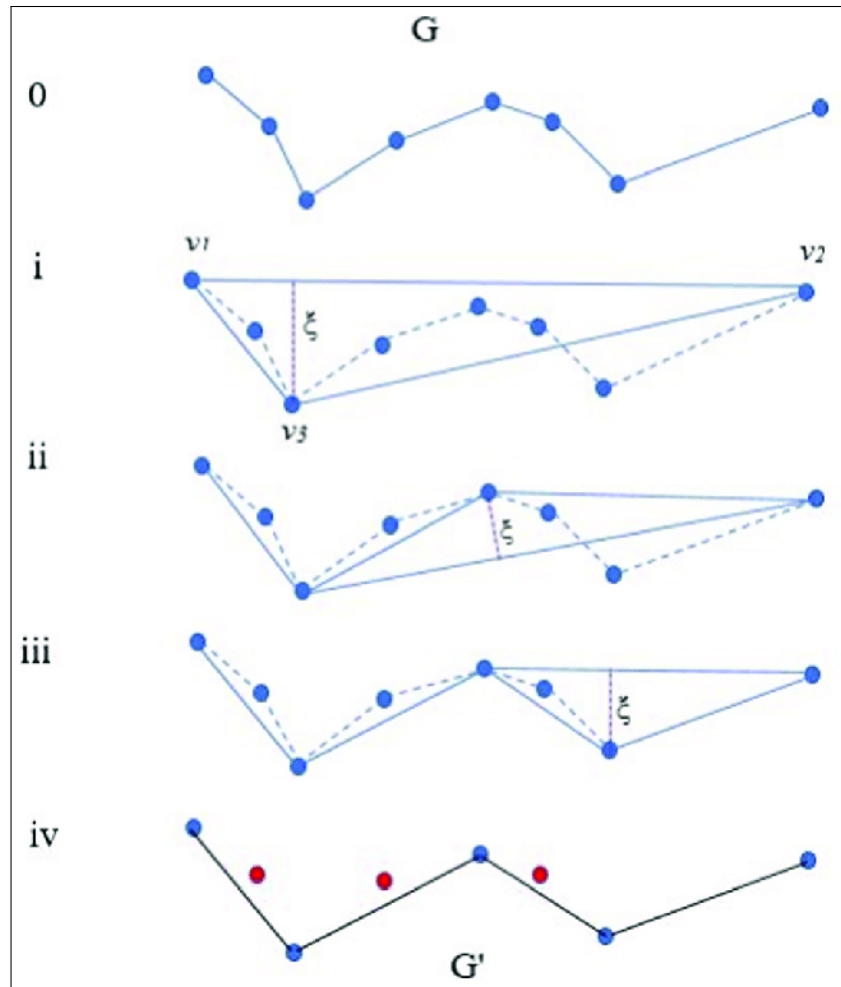


Figura 19: Representación algoritmo de Ramer-Douglas-Peucker
Fuente: [Saalfeld, 1999].

En el contexto de la generación de ruta, los objetivos de aplicar este algoritmo a las rutas son:

1. Evitar subsecciones innecesarias dentro de la ruta provocadas por la naturaleza discreta de la cuadrícula modelada.
2. Minimizar los zigzag o “dientes de sierra” en las secciones más curvas de la ruta, optimizando así la suavidad y la eficiencia del trayecto.
3. Minimizar la cantidad de nodos que componen la ruta. Dado que el plano está modelado en forma de cuadrícula, existen subsecciones dentro de la ruta con nodos que no añaden información a la ruta.

La elección de ϵ es crucial al momento de simplificar la ruta, ya que un valor muy bajo generaría que el algoritmo identifique una gran parte de los nodos como significativos, provocando

que la simplificación sea muy sutil y la ruta final sea muy similar a la original. Por otro lado, un valor de ϵ muy alto provocaría una sobre-simplificación de la ruta, perdiendo precisión y la trayectoria deseada. Para determinar el valor óptimo de ϵ , se siguió una estrategia empírica basada en los siguientes pasos:

1. Se consideró que el plano está modelado como una cuadrícula de $2[m] \times 2[m]$.
2. Se buscó eliminar los zigzags o “dientes de sierra” y simplificar la ruta sin perder su esencia.
3. Se realizó una serie de simulaciones con distintos valores de ϵ en varios escenarios dentro del plano utilizado.
4. Se evaluó visualmente y cuantitativamente el grado de simplificación y la preservación de la trayectoria deseada para cada valor de ϵ .
5. Basándose en los resultados de las simulaciones, se determinó que un valor de ϵ igual a $5[m]$ cumple los objetivos de mantener la forma de la ruta y minimizar la cantidad de imperfecciones en la ruta.

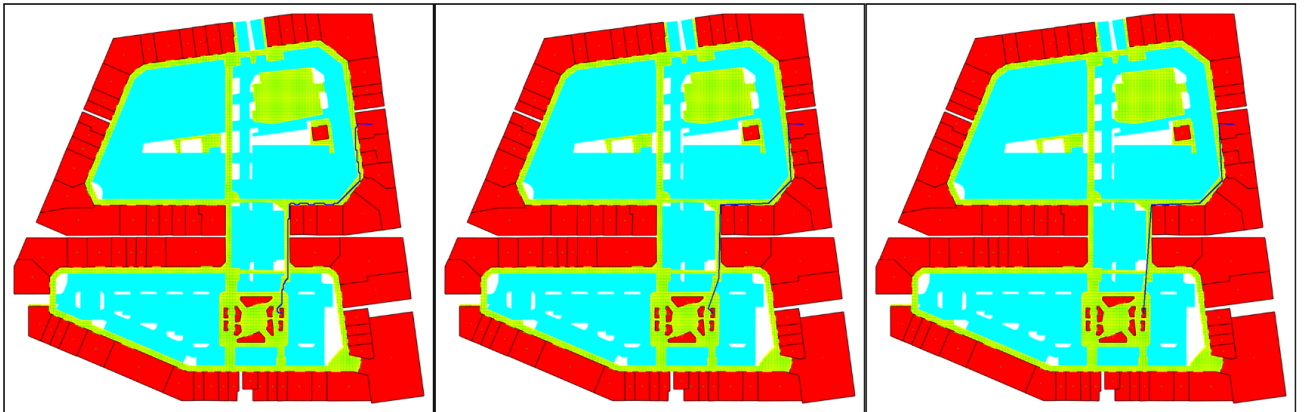


Figura 20: Comparación de rutas optimizadas utilizando el algoritmo de Ramer–Douglas–Peucker con diferentes valores de ϵ . De izquierda a derecha: (a) $\epsilon = 0,5[m]$, resultando en 39 nodos y una ruta muy detallada; (b) $\epsilon = 5[m]$, produciendo 8 nodos y una simplificación moderada; (c) $\epsilon = 10[m]$, generando 6 nodos y una simplificación más agresiva. Se observa cómo el incremento de ϵ reduce la complejidad de la ruta a costa de perder algunos detalles de la trayectoria original.

Fuente: Elaboración propia.

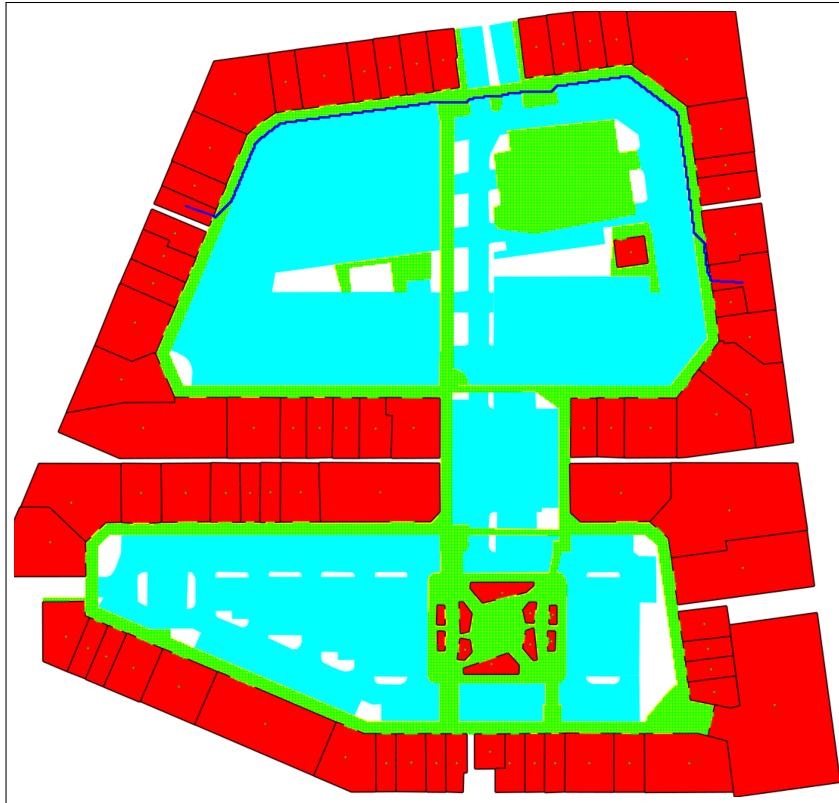


Figura 21: Ruta generada (en azul) utilizando algoritmo A* y distancia Chebyshev como función heurística.

Fuente: Elaboración propia.

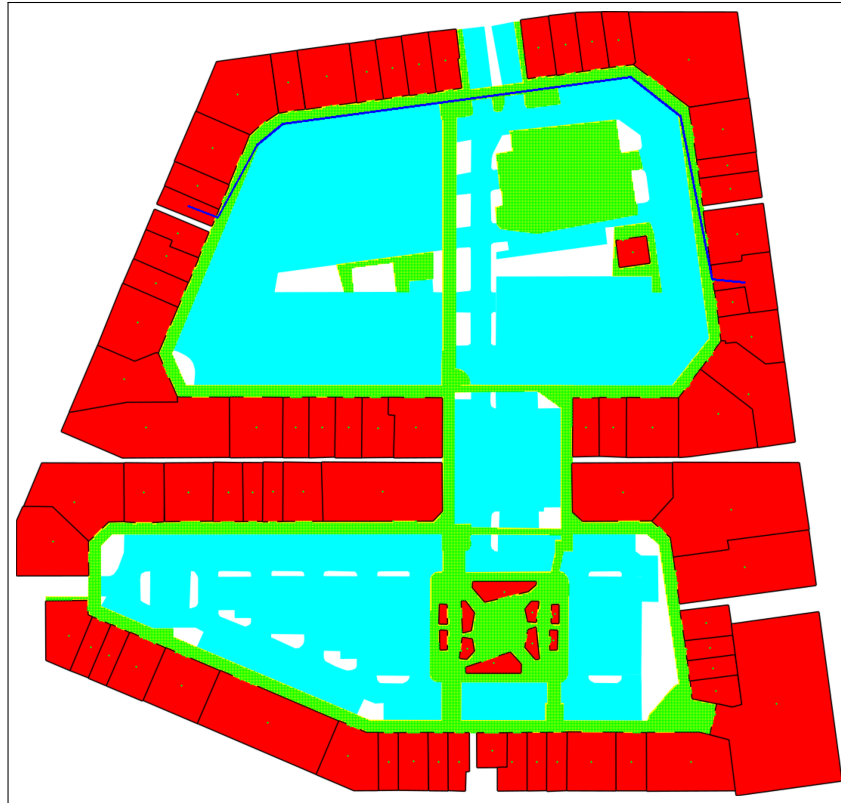


Figura 22: Ruta generada (en azul) con algoritmo A* y optimizada con algoritmo de Ramer-Douglas-Peucker

Fuente: Elaboración propia.

Una vez aplicado el algoritmo de Ramer-Douglas-Peucker a una ruta de ejemplo, es posible apreciar que la ruta original se simplificó evitando movimientos innecesarios y manteniendo la trayectoria original. De manera mas específica, la ruta disminuyó la cantidad de nodos que la componen en un 98,19 %, en donde originalmente se componía de 442 nodos distintos y resultando en una ruta simplificada de tan solo 8 nodos.

Algoritmo	Principio de Funcionamiento	Características Principales
A* con distancia de Chebyshev	Utiliza la distancia de Chebyshev como función heurística	■ Genera rutas más naturales y directas ■ Favorece movimientos ortogonales o diagonales directos ■ Útil en escenarios cuadriculados con movimientos diagonales
Ramer-Douglas-Peucker	Simplifica una curva reduciendo el número de puntos	■ Traza una recta entre puntos inicial y final ■ Selecciona puntos significativos basados en distancia a la recta ■ Aplica recursivamente a segmentos generados ■ Utiliza un umbral ϵ para determinar la simplificación

Tabla 5: Comparación de algoritmos de optimización de rutas

3.4. Aplicación Móvil

El desarrollo de la aplicación móvil constituye una parte fundamental de este proyecto, cuyo objetivo es proporcionar una solución práctica y accesible para obtener información relevante en espacios concurridos. Esta sección detalla el proceso de desarrollo, y las tecnologías utilizadas para crear una aplicación que cumpla con los requisitos establecidos.

3.4.1. Tecnologías Utilizadas

Como se mencionó en el marco teórico y dado el enfoque de desarrollar una aplicación para uso público, se seleccionó React Native como el framework para el desarrollo de la aplicación móvil. React Native es un framework enfocado en el desarrollo móvil híbrido, es decir, que con el mismo código fuente es posible exportar la aplicación a diferentes sistemas operativos, específicamente Android e iOS. Además se utilizó Expo, el cual es un paquete que extiende las funcionalidades de React Native y facilita el proceso de desarrollo.

3.4.2. Interfaz de Usuario

Para la visualización de los mapas, se utilizó el plano procesado manualmente que se presentó previamente. Dado que este plano cuenta con las diferentes zonas identificadas, tales como las tiendas, zonas transitables, estacionamientos y paredes, es posible extraer los contornos y construir una representación 3D que se asemeje a la realidad. Para esto, se extrajo el contorno de cada zona distinta y se representó en 3D utilizando el paquete de código abierto para crear y visualizar modelos 3D en la web. De esta manera, a partir del contorno 2D del lugar de ejemplo, se le asigna una altura aproximada, dependiendo de la zona, y se genera un plano 3D el cual facilita la navegación y hace más atractiva la aplicación.

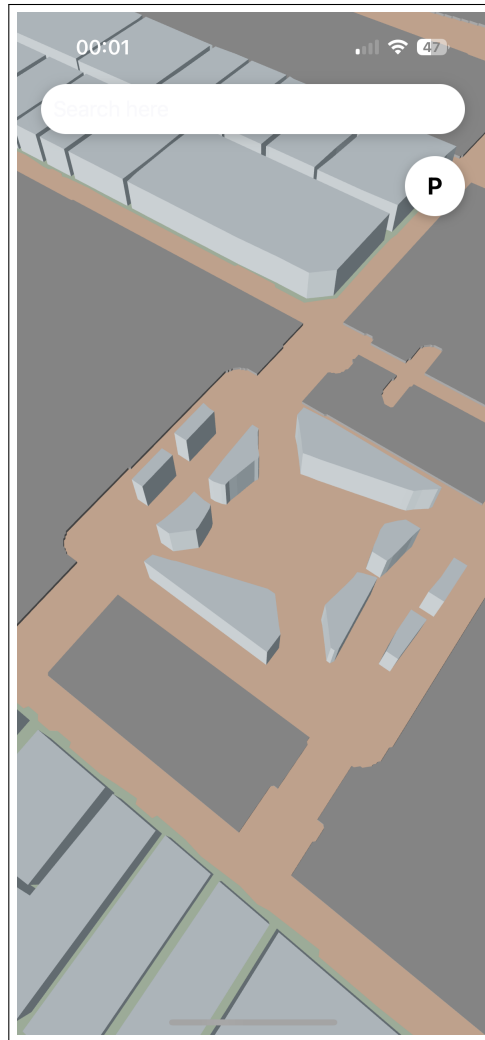


Figura 23: Visualización de mapa 3D utilizando contornos y Three.JS.
Fuente: Elaboración propia.

Por otro lado, se añadió una barra de búsqueda en la zona superior de la pantalla para facilitar la búsqueda de tiendas en el centro comercial de ejemplo. Una vez seleccionada la tienda a visualizar, se enfoca en color rojo y se visualiza información relevante relacionada a esta, junto a un botón para obtener la ruta hasta la tienda seleccionada. Para mostrar esta información, se utilizó el paquete de código abierto *@gorhom/bottom-sheet*, el cual ofrece una pestaña desplazable y versátil.

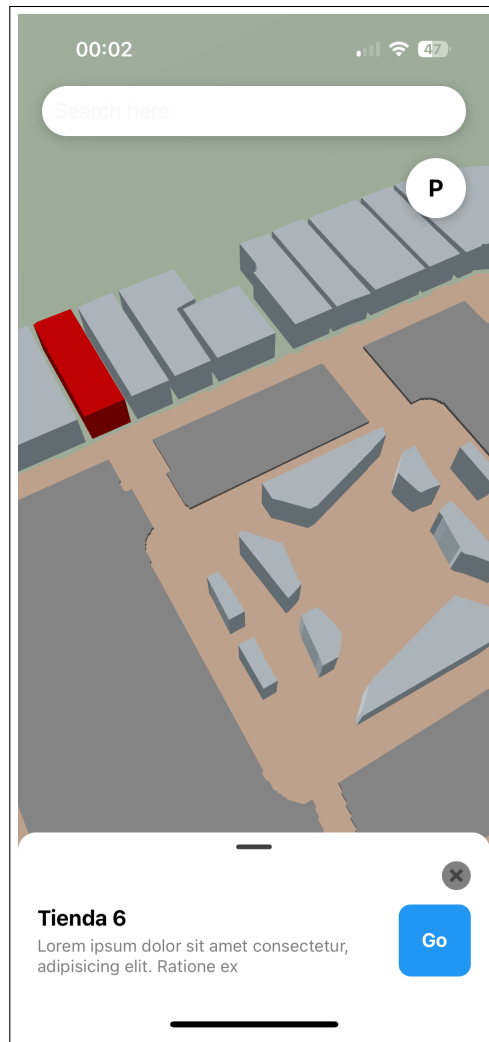


Figura 24: Visualización de tienda e información relevante.
Fuente: Elaboración propia.

Finalmente, para visualizar la ruta generada una vez que seleccionada, se utilizó una forma de sección tubular propia del paquete Three.JS. La ruta generada muestra claramente los puntos iniciales y finales, y es posible seguirla durante todo el trayecto.

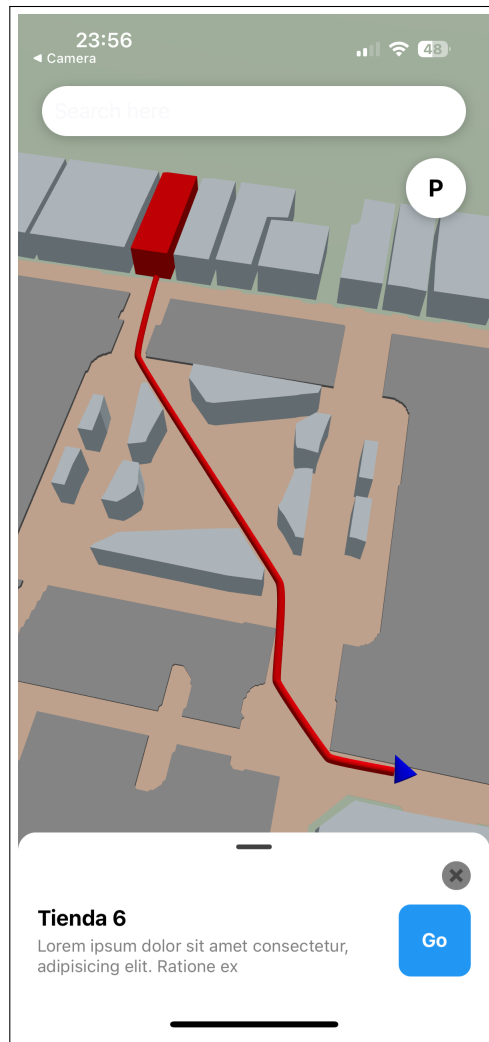


Figura 25: Visualización de ruta utilizando Three.JS.
Fuente: Elaboración propia.

3.5. Servidor de Backend

Para el procesamiento de datos proveniente del controlador de Cisco Meraki y proveer información de la ubicación en tiempo real, se construyó un servidor de backend utilizando Node.JS, el cual es un entorno de ejecución de JavaScript ampliamente utilizado en el desarrollo web. Además se utilizó el paquete Express.JS, el cual es un paquete escrito en JavaScript que proporciona herramientas y características para crear un servidor API Rest de forma sencilla. Este servidor proporciona dos servicios importantes para el funcionamiento de la aplicación:

3.5.1. Conexión con Cisco Meraki

Para obtener la información relacionada a la ubicación de los dispositivos conectado a una red, se debe configurar el controlador de Cisco Meraki. En este controlador es posible configurar un webhook que enviará constantemente la información de los dispositivos detectados, junto a información como su dirección IP, su dirección MAC y ubicación aproximada. Por lo tanto, Cisco Meraki envía la información actual en formato JSON a una ruta en el servidor backend. Una vez recibido por el servidor, este se encarga de eliminar los registros sin ubicación aproximada, los agrupa por dirección IP y se guardan en una base de datos relacional PostgreSQL para obtenerla posteriormente. Debido al alto flujo de datos proveniente del controlador, la información con mas de 24 horas de antigüedad es eliminada para que las consultas a la base de datos no se vuelvan mas costosas.

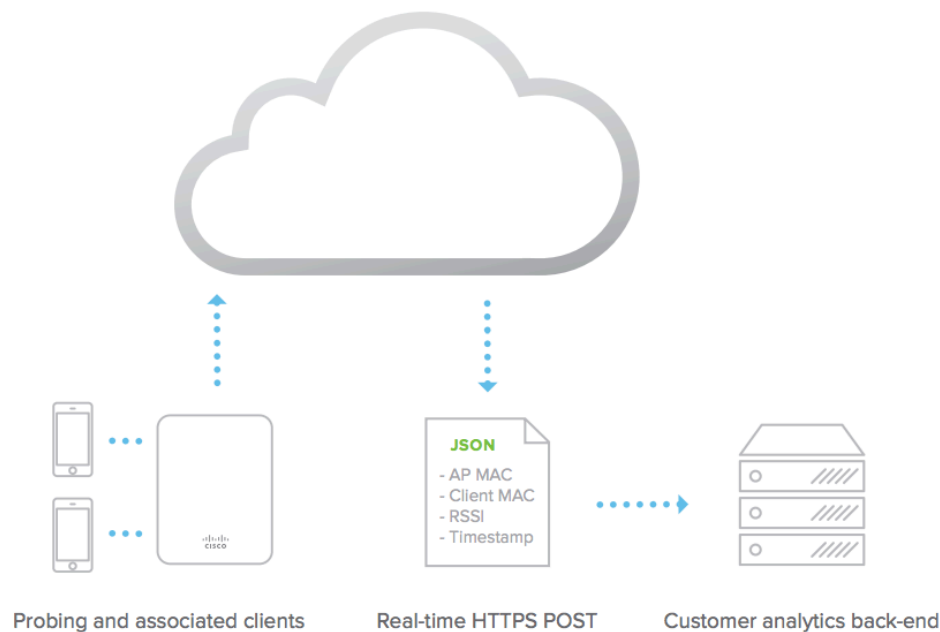


Figura 26: Diagrama del funcionamiento de la API de Location Analytics de Cisco Meraki.
Fuente: [Meraki, 2024].

3.5.2. Obtención de Ubicación

Una vez que un usuario abre la aplicación y esta conectado a la red Wi-Fi correcta, se envía al servidor una petición GET para obtener la ubicación de la dirección IP asociada al dispositivo. Una vez recibido, el servidor obtiene las ultimas ubicaciones guardadas en la base de datos y a partir de ellas, aplica un modelo que aproxima la ubicación actual en base a las anteriores, con el fin de minimizar los errores. Luego se envía al dispositivo para mostrarlo en pantalla.

3.5.3. Obtención de rutas

Cuando un usuario selecciona una tienda en la aplicación móvil y desea ver la ruta desde la posición actual, se envía al servidor una petición GET para obtener la ruta desde la posición actual a la tienda objetivo. Entonces, dentro del servidor se ejecutan los algoritmos mencionados previamente para generar una ruta óptima, visualmente acertada y simplificada, para finalmente poder visualizarla en la aplicación móvil.

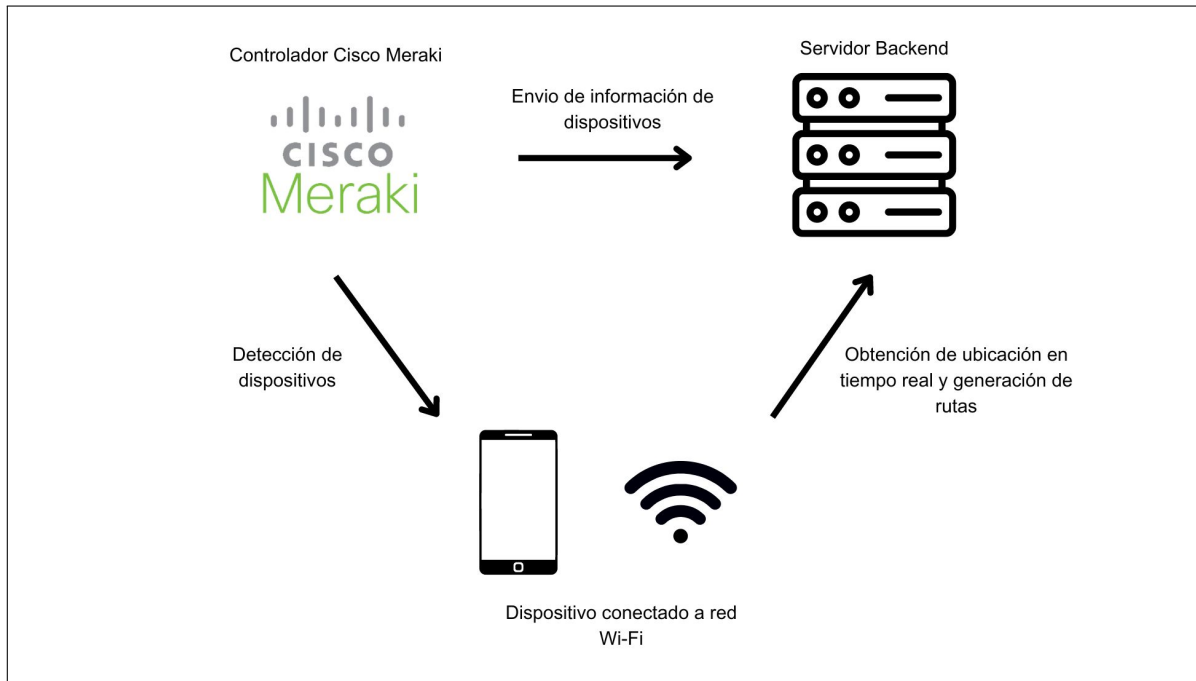


Figura 27: Diagrama funcionamiento de obtención de rutas.
Fuente: Elaboración propia.

3.6. Aproximación de Ubicación en Tiempo Real

Una vez obtenida la ubicación en tiempo real de los dispositivos detectados por la red de Cisco Meraki, se graficaron los datos para realizar una comparación visual con la ruta originalmente planificada.

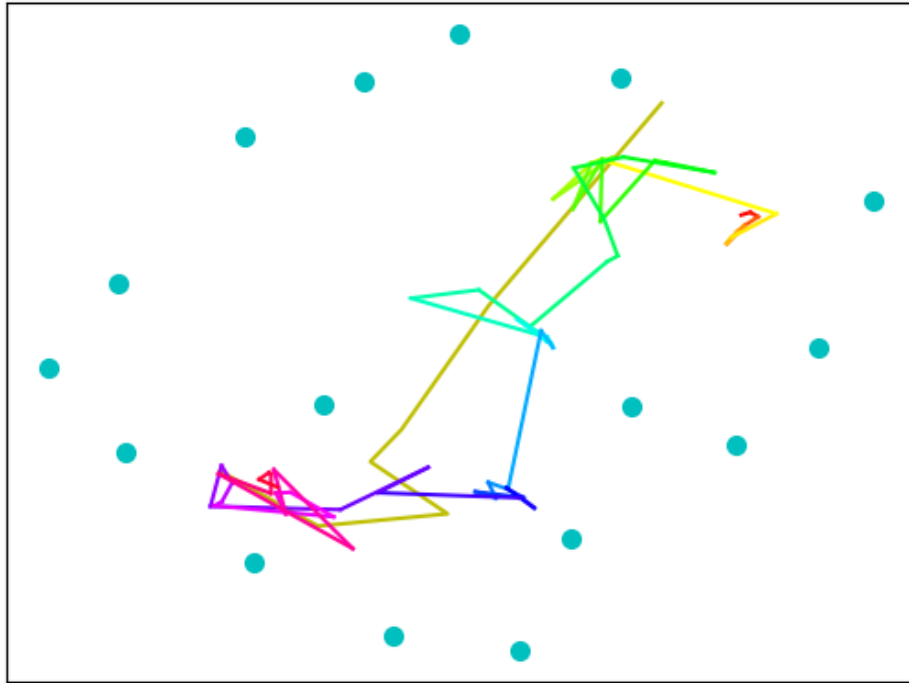


Figura 28: Ruta original (en amarillo) junto a la ruta calculada por la red de Cisco Meraki (en multicolor).

Fuente: Elaboración propia.

En el gráfico, se puede observar que la ruta obtenida desde la red de Cisco Meraki no presenta una gran semejanza con la ruta original planificada. Por lo tanto, se buscaron alternativas para mejorar la detección de la ubicación en tiempo real.

3.6.1. Teoría del Filtro de Kalman

Se decidió utilizar el filtro de Kalman, un algoritmo recursivo que estima el estado de un sistema dinámico a partir de mediciones ruidosas. El filtro de Kalman opera en dos fases principales:

1. Predicción: Estima el estado actual basándose en el estado anterior.
2. Actualización: Corrige la estimación utilizando las mediciones actuales.

Las ecuaciones principales del filtro de Kalman son:

Predicción:

$$\begin{aligned}\hat{x}_k^- &= F_k \hat{x}_{k-1} + B_k u_k \\ P_k^- &= F_k P_{k-1} F_k^T + Q_k\end{aligned}$$

Actualización:

$$\begin{aligned}K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k (z_k - H_k \hat{x}_k^-) \\ P_k &= (I - K_k H_k) P_k^-\end{aligned}$$

Donde:

- $\hat{x}_k$ es el estado estimado
- P_k es la matriz de covarianza del error de estimación
- F_k es la matriz de transición de estado
- H_k es la matriz de observación
- Q_k es la matriz de covarianza del ruido del proceso
- R_k es la matriz de covarianza del ruido de medición
- K_k es la ganancia de Kalman

3.6.2. Implementación

Para la implementación, se utilizó el paquete FilterPy de Python. Se inicializó el filtro de Kalman con un estado de dimensión 4 (posiciones X e Y, y velocidades en ambos ejes) y mediciones de dimensión 2 (latitud y longitud).

```
kf = KalmanFilter(dim_x=4, dim_z=2)
```

Se inicializaron las matrices necesarias:

```
kf.H = np.array([[1, 0, 0, 0],
                 [0, 0, 1, 0]])
kf.Q = np.eye(4) * Q
kf.P *= P
initial = measurements[0]
kf.x = np.array([initial['lat'], 0, initial['lng'], 0])
```

En cada iteración, se actualiza la matriz de transición de estado F y la matriz de covarianza de medición R:

```
kf.F = np.array([[1, dt, 0, 0],
                 [0, 1, 0, 0],
                 [0, 0, 1, dt],
                 [0, 0, 0, 1]])
kf.R = np.diag([measurement['unc']**R, measurement['unc']**R])
```

3.6.3. Ajuste de Parámetros

Los parámetros clave del filtro de Kalman (Q, R, y P) se ajustaron mediante un proceso de búsqueda exhaustiva para optimizar el rendimiento del filtro. Se probaron los siguientes rangos:

- Q: np.logspace(-6, 2, 20)
- P: np.linspace(1, 1000, 20)
- R: np.linspace(0, 12, 20)

Se evaluó el rendimiento utilizando la distancia euclidiana paso-a-paso entre la ruta predicha y la ruta real. Los valores óptimos encontrados fueron:

- Q: 0.05455594
- P: 1
- R: 2.6

Estos valores proporcionaron el mejor equilibrio entre la suavidad de la trayectoria estimada y la capacidad de seguir los cambios reales en la posición.

3.6.4. Ajuste a Restricciones Físicas

Para mejorar la precisión en el contexto del espacio físico, se implementó una verificación adicional:

```
if not is_transitable(lat, lng):
    new_lat, new_lng = closest_transitable(lat, lng)
    kf.x[0], kf.x[2] = new_lat, new_lng
```

Esta verificación asegura que las posiciones estimadas siempre correspondan a áreas transitables del espacio, mejorando la coherencia de las predicciones con la realidad física del entorno.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

La validación de la solución se lleva a cabo a través de varias etapas, incluyendo la comparación técnica de algoritmos de rutas, pruebas de rendimiento del filtro de Kalman y pruebas de la aplicación final. A continuación, se presentan los detalles de cada etapa.

4.1. Evaluación de algoritmos de rutas

Inicialmente, para obtener rutas óptimas entre dos puntos del espacio modelado se revisó varios algoritmos populares, entre los cuales se encuentran el algoritmo de Dijkstra, el algoritmo Bellman-Ford y el que se seleccionó finalmente, es algoritmo A*. Estos algoritmos son ampliamente conocidos y utilizados, sin embargo, el algoritmo A* tiene características beneficiosas para el caso de uso. Entre las características de cada algoritmo se encuentra.

Característica	Dijkstra	Bellman-Ford	A*
Uso principal	Camino más corto desde un nodo origen hasta los demás nodos, donde las aristas tienen pesos positivos	Camino más corto desde un nodo origen hasta los demás nodos, permitiendo aristas con pesos negativos	Camino más corto desde un nodo origen hasta el nodo destino utilizando heurística
Complejidad Temporal	$O((V + E)\log V)$	$O(VE)$	$O((V + E)\log V)$
Heurística	No utiliza	No utiliza	Utiliza

Tabla 6: Comparación de Algoritmos de Ruta, donde V son el numero de vértices y E el numero de aristas del grafo.

En una comparación teórica, los algoritmos de Dijkstra y A* son los menos complejos temporalmente, lo cual es beneficioso en un escenario en donde se necesita manipular datos constantemente para obtener la ubicación en tiempo real. Por otro lado, la complejidad temporal entre Dijkstra y A* depende totalmente de la heurística utilizada en esta ultima. En el contexto del algoritmo A*, la función heurística determina el costo de ir desde el nodo actual hasta el nodo objetivo. Si la función heurística sobrestima el costo real, podría causar ineficiencia en la ejecución y aumentar la complejidad temporal. Sin embargo, la utilización de heurísticas es un beneficio al momento de querer personalizar la forma en que se calcula el costo de ir de un nodo a otro. Por ejemplo, es posible implementar el algoritmo A* con heurísticas para minimizar los movimientos diagonales, para minimizar los cambios de dirección, etc; dependiendo del objetivo específico.

4.1.1. Comparación técnica de algoritmos

Para comparar los algoritmos, se implementaron los algoritmos utilizando Python 3.10.13 (Aug, 2023) y fueron ejecutadas en un dispositivo Macbook Pro con un procesador Apple M1 y 16GB de memoria RAM.

Utilizando el plano procesado y el grafo generado, se utilizaron implementaciones estándar de los algoritmos Dijkstra y A*, este último utilizando como heurística la distancia de Chebyshev. Luego, se ejecutaron 200 iteraciones en donde inicialmente se obtenía un punto inicial y un punto objetivo al azar, para luego ejecutar ambos algoritmos utilizando los mismos puntos.

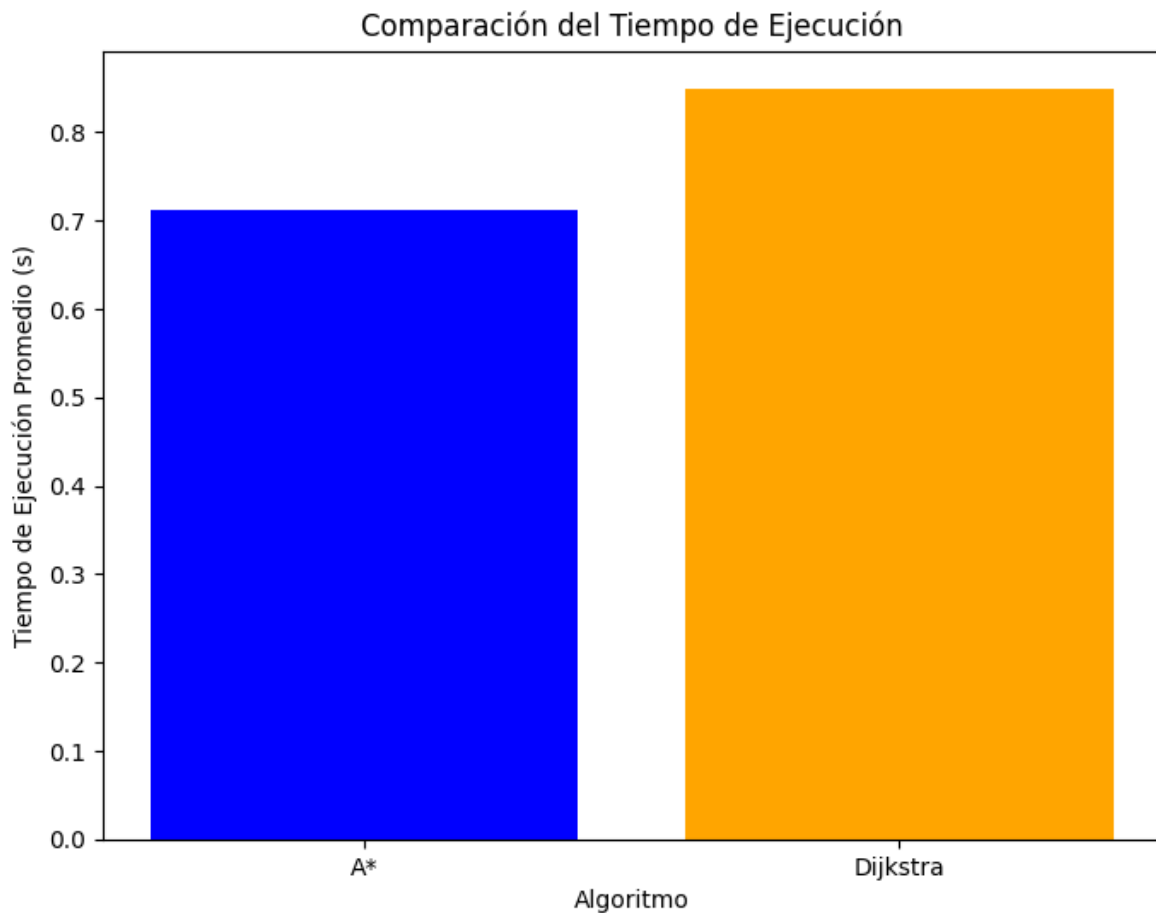


Figura 29: Comparación tiempo de ejecución promedio.
Fuente: Elaboración propia.

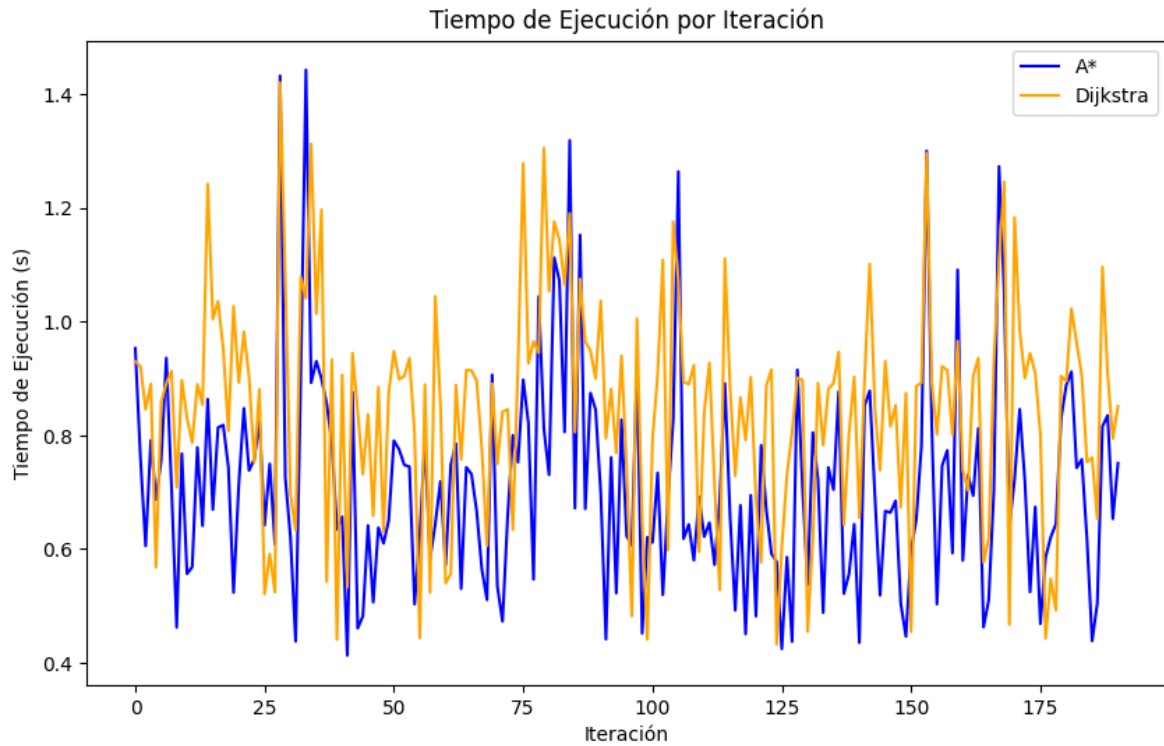


Figura 30: Comparación tiempo de ejecución por iteración.
Fuente: Elaboración propia.

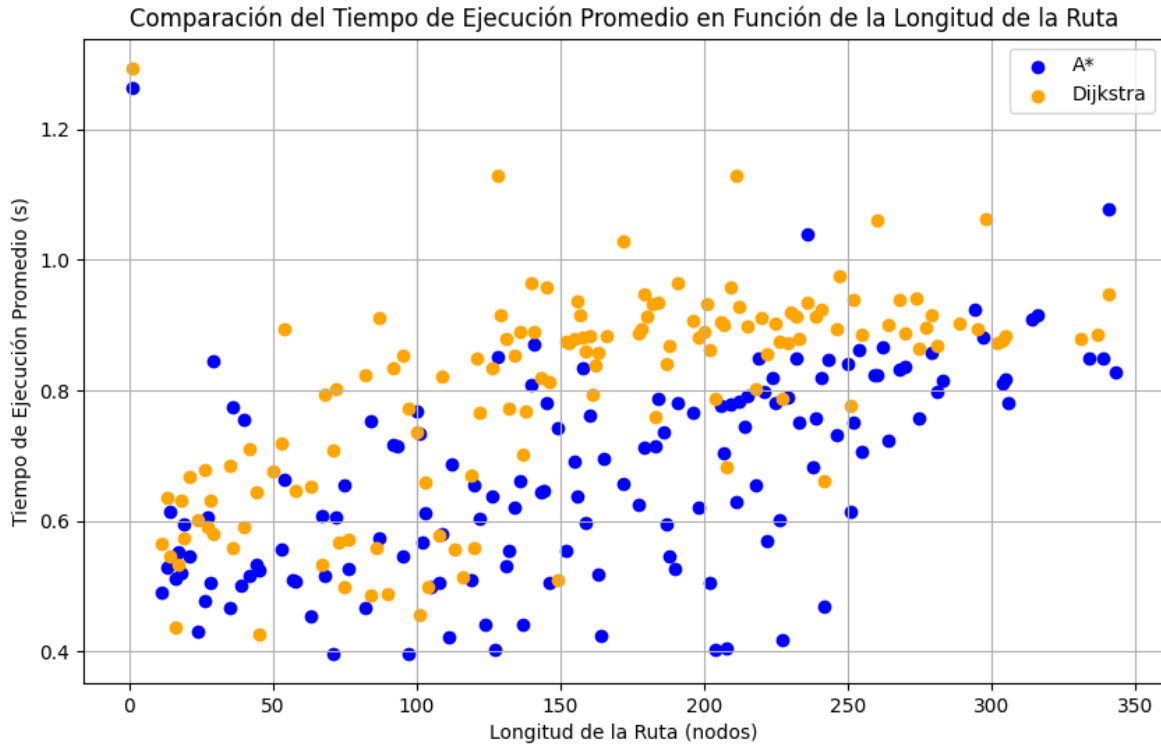


Figura 31: Gráfico de tiempo de ejecución promedio en función de la longitud de la ruta.
Fuente: Elaboración propia.

En promedio, el algoritmo de Dijkstra es más de 100 milisegundos más costoso temporalmente que el algoritmo A* utilizando la distancia de Chebyshev como función heurística. Además, en la mayoría de los casos, el algoritmo A* tarda menos en ejecutarse en comparación con el algoritmo de Dijkstra, con diferencias de hasta 0.58 segundos (iteración 102). Sin embargo, cuando las rutas generadas están compuestas por un número reducido de nodos, ambos algoritmos se comportan de manera similar. A medida que los caminos se vuelven más extensos, el algoritmo A* se vuelve menos costoso en términos de tiempo de ejecución, esto debido a los métodos de búsqueda que utilizan ambos algoritmos. Mientras que el algoritmo de Dijkstra realiza una expansión en anchura basada en el costo acumulado, el algoritmo A* utiliza una heurística para ir directamente hacia el nodo objetivo y no visitar nodos innecesarios.

4.1.2. Comparación visual de algoritmos

Por otro lado, es importante comparar que tan preciso es cada algoritmo al momento de generar rutas entre un punto inicio y uno objetivo en cualquier parte del plano. Al momento de generar los planos usando cualquiera de ambos algoritmos, se aplicó una simplificación de la ruta generada utilizando el algoritmo de Ramer-Douglas-Peucker, el cual simplifica una

curva minimizando la cantidad de puntos que la componen y que sea aproximadamente igual a la original.

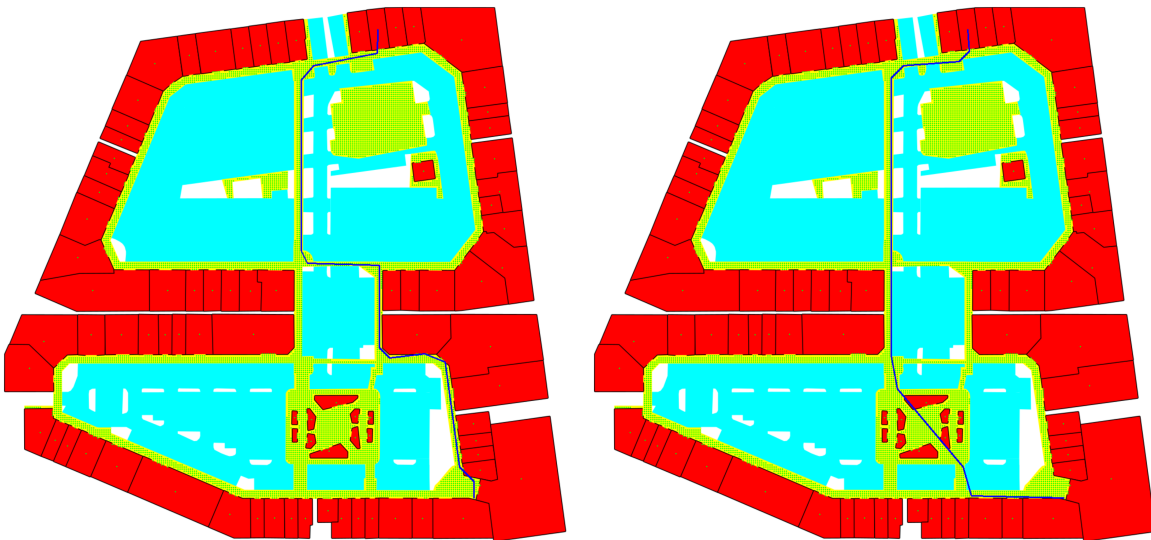


Figura 32: Rutas extensas generadas con Algoritmo A* (Izquierda) versus Algoritmo de Dijkstra (Derecha), ambas simplificadas aplicando algoritmo de Ramer-Douglas-Peucker.

Fuente: Elaboración propia.

Comparando rutas extensas generadas con ambos algoritmos, es posible notar que ambos generaron una ruta inicialmente similar, pero que cuando llegaron a zonas donde existen distintas opciones, ambos eligieron distintas formas de llegar al objetivo.

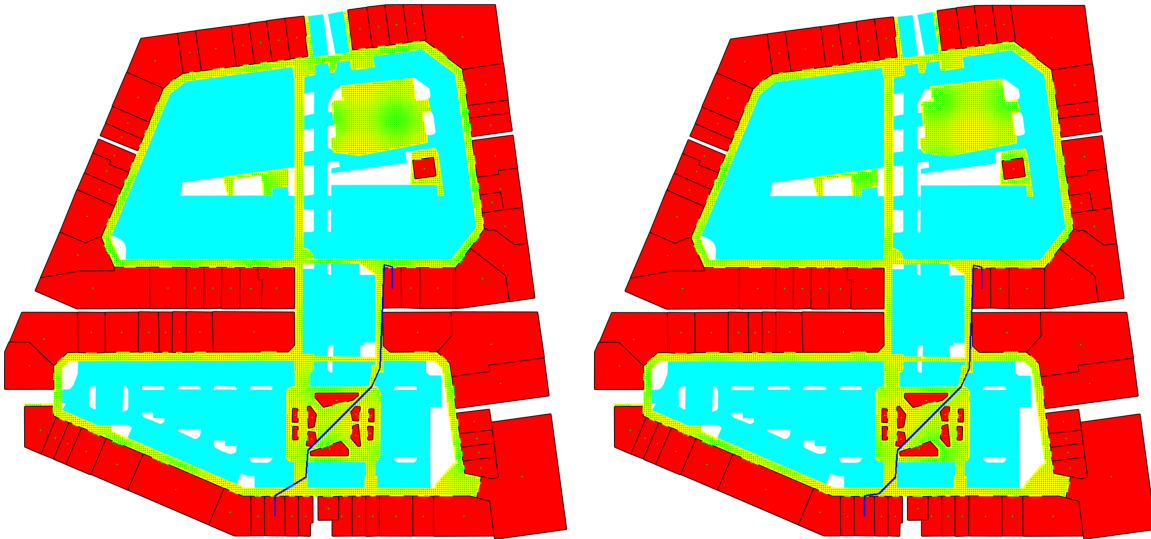


Figura 33: Rutas con obstáculos utilizando Algoritmo A* (Izquierda) versus Algoritmo de Dijkstra (Derecha), ambas simplificadas aplicando algoritmo de Ramer-Douglas-Peucker.

Fuente: Elaboración propia.

Las figuras anteriores muestran que las rutas generadas por A* tienden a ser más directas y eficientes en comparación con las generadas por Dijkstra. Esto se debe a la utilización de heurísticas en el algoritmo A*, que guían la búsqueda hacia el objetivo de manera más efectiva. En contraste, Dijkstra evalúa todas las rutas posibles sin una guía específica, lo que puede resultar en caminos menos óptimos y mayores tiempos de ejecución.

4.2. Filtrado de Kalman

Como se explicó en la sección anterior, el filtro de Kalman es un algoritmo enfocado en obtener el estado óptimo de un sistema dinámico a partir de mediciones previas. Para esto, se realizó una implementación modificada del algoritmo, en donde las mediciones predichas son aproximadas al nodo transitable mas cercano, con el fin de evitar predecir posiciones en lugares imposibles del entorno, tales como jardines decorativos o espacios no transitables.

Previo a realizar las pruebas de validación de la implementación del filtrado de Kalman es necesario ajustar las variables iniciales del algoritmo, las cuales son:

Matriz Q Matriz de covarianza del ruido del proceso, representa la incertidumbre en el modelo del proceso y especifica el ruido esperado en la transición del estado. Depende de la dinámica del sistema y la confianza en el modelo del proceso. Un valor alto de Q indica mucha incertidumbre en el modelo, haciendo que el filtro de Kalman confíe más en las mediciones.

Matriz P Matriz de covarianza del error de estimación inicial. Representa la incertidumbre en el estado inicial estimado. Inicialmente, indica cuánta confianza tenemos en nuestra estimación inicial. Un valor alto de P indica que hay mucha incertidumbre en la estimación inicial.

Matriz R Matriz de covarianza del ruido de medición. Representa la incertidumbre en las mediciones y cuánto ruido esperamos.

Para esto, se realizó un proceso de búsqueda exhaustiva de los parámetros óptimos para el espacio que se está analizando. Para lo cual, se definieron intervalos de posibles valores para las variables antes mencionadas. La matriz de covarianza del ruido del proceso Q se estableció con valores definidos en $np.logspace(-6, 2, 20)$, lo cual genera una serie de valores espaciados uniformemente en una escala logarítmica desde 10^{-6} hasta 10^2 . Esta elección permite explorar una amplia gama de magnitudes, capturando tanto valores muy pequeños (indicando baja incertidumbre) como valores grandes (indicando alta incertidumbre), lo que es esencial para entender cómo diferentes niveles de ruido del proceso afectan el rendimiento del filtro. La matriz de covarianza del error estimado inicial P se definió como $np.linspace(1, 1000, 20)$, es decir, una serie de valores uniformemente espaciados de manera lineal entre 1 y 1000. Esto permite evaluar el impacto de diferentes niveles de confianza en la estimación inicial del estado, desde muy confiable (valores bajos) hasta muy incierto (valores altos). Finalmente, la matriz de covarianza del ruido de medición R , que en la implementación se define como una matriz $np.diag([measurement['unc']^R, measurement['unc']^R])$, se elevó a valores definidos según $np.linspace(0, 12, 20)$, es decir, valores espaciados uniformemente de manera lineal entre 0 y 12. Este rango permite explorar cómo diferentes niveles de incertidumbre en las mediciones afectan la precisión del filtro, desde ninguna incertidumbre adicional ($R = 0$) hasta niveles muy altos de incertidumbre ($R = 12$). Explorar distintos valores de R nos permite observar el efecto de penalizar más las mediciones, proporcionando una comprensión más profunda de la sensibilidad del filtro a la incertidumbre de las mediciones.

Una vez definidos los posibles valores de las variables antes mencionadas, se seleccionó una métrica la cual nos permita comparar la validez y semejanza entre la ruta original y la ruta predicha utilizando el filtrado de Kalman. Con este objetivo, se seleccionó la distancia euclidiana paso-a-paso (lock-step Euclidean distance), la cual es una métrica que calcula la distancia euclidiana entre pares consecutivos de puntos de dos rutas distintas. Esta métrica se utilizó en un proceso iterativo para encontrar los valores de Q , P y R que minimicen este valor con el objetivo de encontrar la ruta mas similar a la realidad. Luego, se ejecutó una prueba de búsqueda exhaustiva para encontrar los valores de los parámetros utilizados en el filtrado de Kalman con el fin de encontrar una predicción de la posición lo mas similar a la realidad. Para esto, se iteró sobre todas las combinaciones posibles de los parámetros hasta encontrar las predicciones que minimizaran la distancia euclidiana paso-a-paso entre la ruta predicha y la original. Al momento de hacer las pruebas, el input inicial fueron las posiciones de un dispositivo detectado en un centro comercial con una red con tecnología Cisco Meraki implementado junto con la ruta hecha realmente.

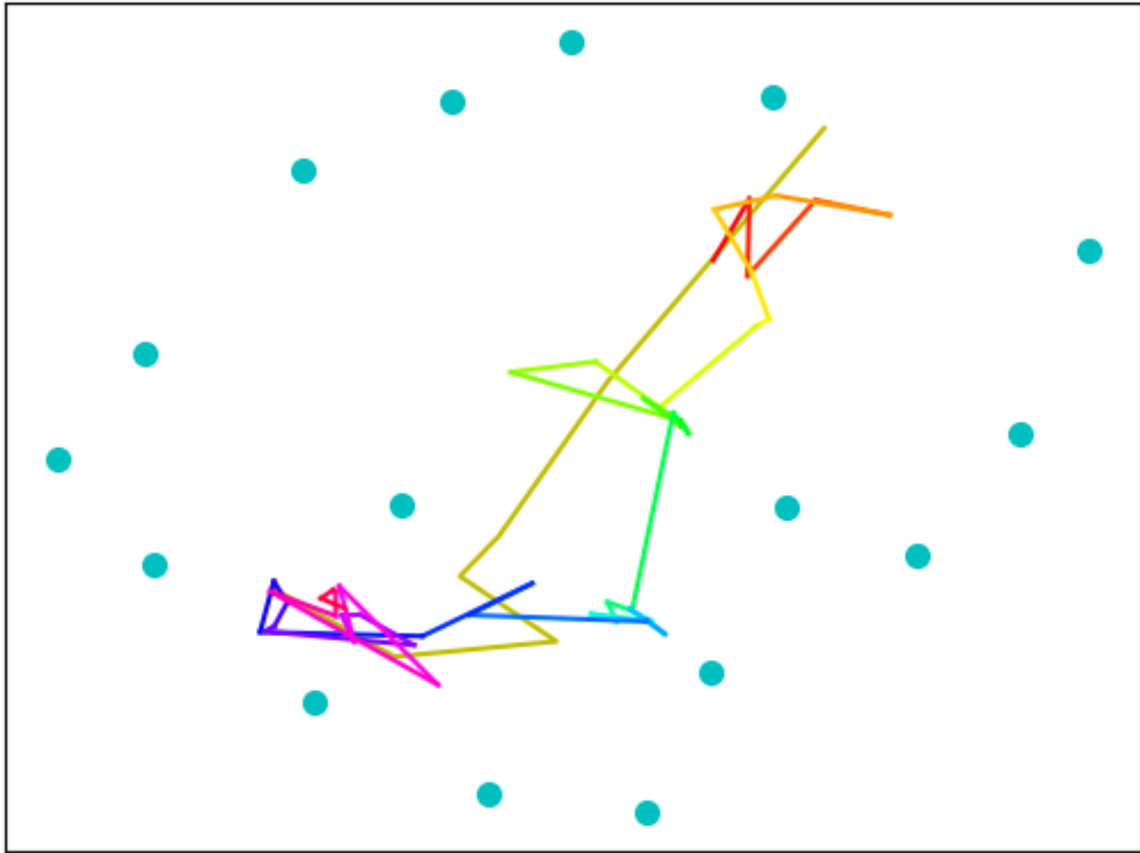


Figura 34: Ubicación de un dispositivo calculada por Cisco Meraki (Multicolor) y ruta real (Amarillo)

Fuente: Elaboración propia.

Luego, se comenzó a iterar sobre las combinaciones de parámetros, se calculó que tan cercano es la posición predicha por el filtro de Kalman en relación a las posiciones originales, hasta encontrar los parámetros que minimizaran este valor. Analizando las miles posibles combinaciones de parámetros, se seleccionó como punto óptimo los valores 0,05455594 para la matriz Q , 1 para la matriz P y 2,6 para la matriz R , obteniendo como resultado una predicción de posiciones mas cercana a la realidad, en donde no hay cambios de dirección extremos ni puntos que van oscilando en ciertas áreas.

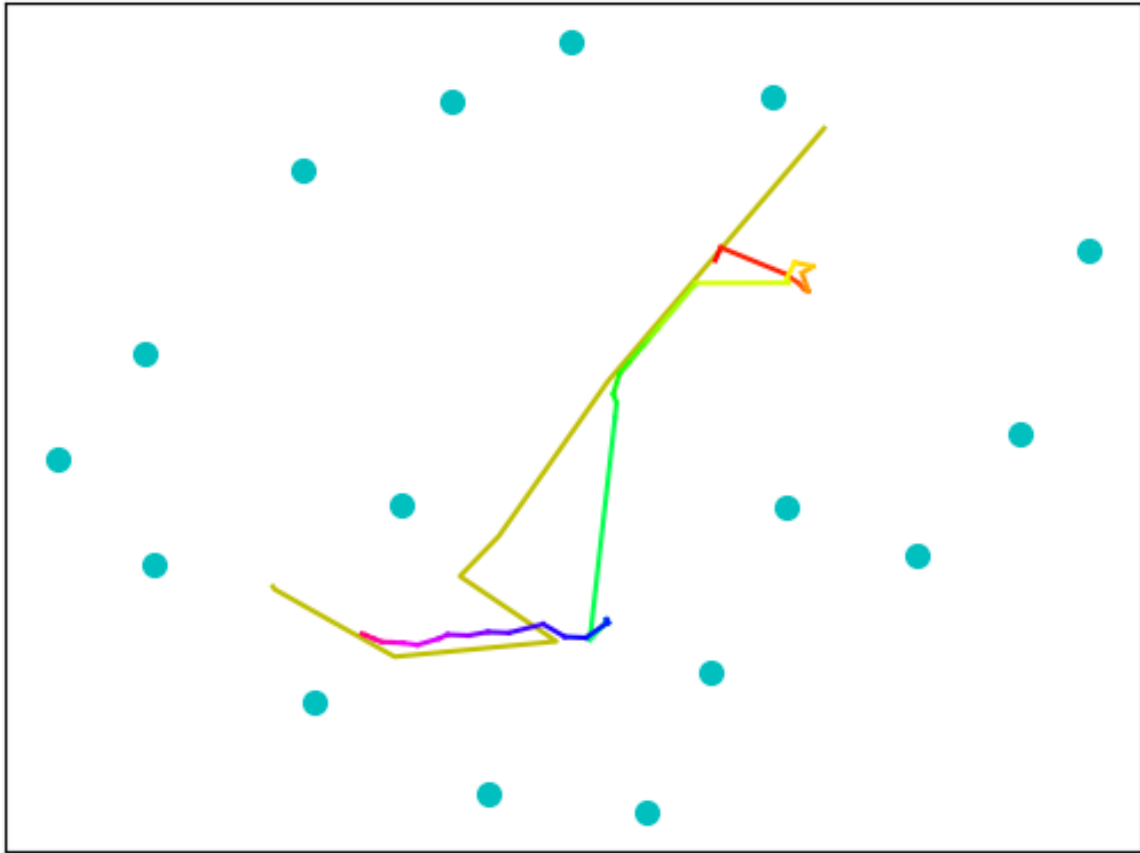


Figura 35: Posiciones predichas utilizando filtro de Kalman (Multicolor) y posiciones reales (Amarillo).

Fuente: Elaboración propia.

Para comparar la ruta realizada realmente con las posiciones predichas utilizando el filtrado de Kalman se graficaron los errores de posición comparando ambas rutas y calculando la diferencia punto a punto. Es importante aclarar que las posiciones obtenidas desde la red de Cisco Meraki no es constante y puede generar huecos de información debido a distintos factores, tales como falta de cobertura de señal Wi-Fi o problemas en la red. Debido a estos huecos, los errores de posición varían a lo largo del tiempo, siendo menores en periodos con menor concentración de datos y altos en periodos donde el sistema es incapaz de detectar a un dispositivo.

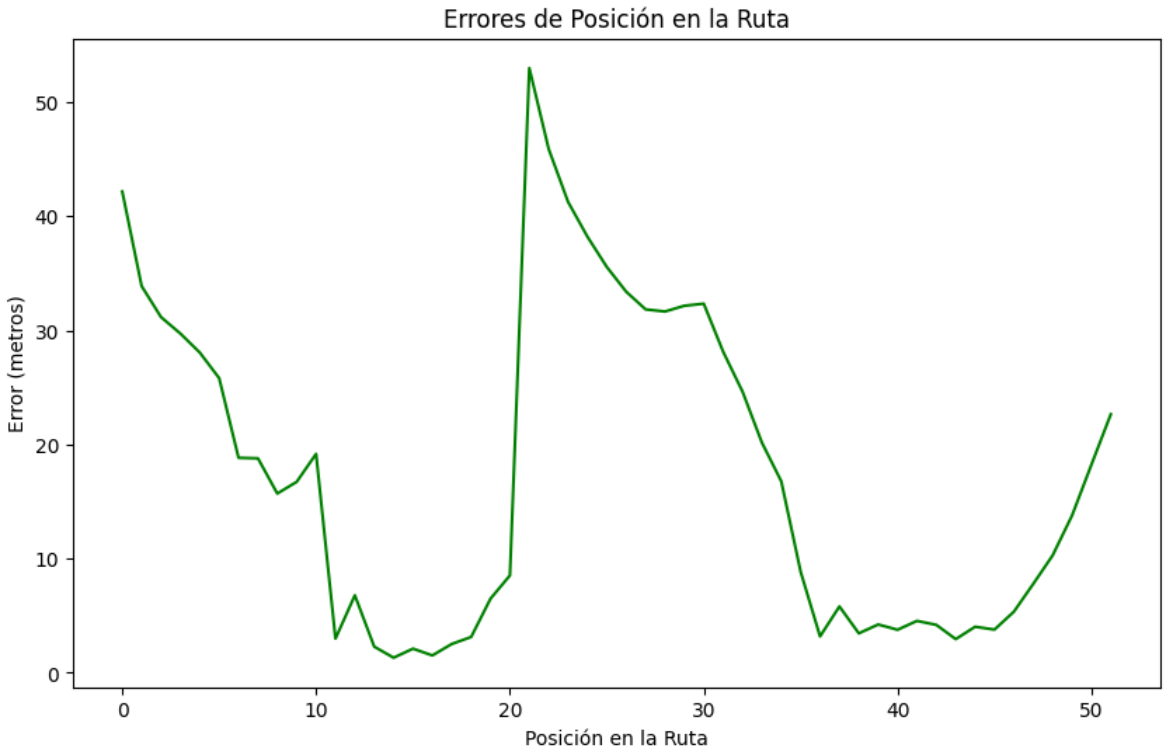


Figura 36: Errores de posición.
Fuente: Elaboración propia.

Por otro lado, la distancia media cuadrática mínima (RMSE) obtenido fue de 23.84 metros, lo que indica la magnitud típica de los errores de predicción. Por otro lado, el MAE resultó en 18.58 metros, representando el error promedio en términos absolutos. Estos valores sugieren que, si bien el filtro de Kalman mejora significativamente la coherencia de la trayectoria, aún existe un margen considerable para mejorar la precisión absoluta de las predicciones. Es importante considerar que estos errores están influenciados por factores como los huecos de información en las mediciones de Cisco Meraki y las limitaciones inherentes al entorno del centro comercial, como la presencia de obstáculos y la variabilidad en la intensidad de la señal Wi-Fi.

Distancia Media Cuadrática Mínima (RMSE)	23.84 [metro]
Error Absoluto Medio (MAE)	18.58 [metro]

CAPÍTULO 5

CONCLUSIONES

Este trabajo de investigación se centró en el desarrollo de una aplicación que aplique posicionamiento interno para la detección de usuarios, representación de puntos de interés y optimización de rutas dentro de espacios cerrados. Los objetivos principales del proyecto fueron:

1. Desarrollar una aplicación móvil para visualización de rutas óptimas y obtención de ubicación en tiempo real.
2. Implementar un sistema de obtención de ubicación actual utilizando tecnologías de localización interna de Cisco Meraki.
3. Generar rutas óptimas dentro de un espacio cerrado.
4. Integrar un modelo 3D con los datos de ubicación en tiempo real.

A través de un proceso de implementación y validación, se han alcanzado varias conclusiones:

1. **Eficacia del algoritmo A*:** La comparación entre los algoritmos Dijkstra y A* demostró la superioridad de este último para la generación de rutas en nuestro contexto. A* no solo resultó ser más eficiente en términos de tiempo de ejecución, especialmente para rutas más largas, sino que también produjo rutas más directas y naturales. Esto valida nuestra elección de A* como el algoritmo principal para la generación de rutas en el sistema, cumpliendo así el tercer objetivo del proyecto.
2. **Optimización de rutas:** La implementación del algoritmo de Ramer-Douglas-Peucker para la simplificación de rutas demostró ser altamente efectiva, logrando una reducción significativa en el número de nodos (98.19 % en el caso de ejemplo) sin comprometer la integridad de la ruta. Optimización importante para reducir la complejidad de las rutas y maximizar la eficiencia del sistema.
3. **Mejora en la estimación de posición:** La aplicación del filtro de Kalman mejoró notablemente la coherencia de las trayectorias estimadas en comparación con los datos brutos proporcionados por la red de Cisco Meraki. Con un RMSE de 23.84 metros y un MAE de 18.58 metros, se logró una mejora importante en la precisión de las posiciones detectadas en comparación a la real, permitiendo un avance para lograr el segundo objetivo, aunque aún hay margen de mejorar.

4. **Integración de modelado 3D:** La implementación exitosa de una representación 3D del espacio, combinada con la visualización en tiempo real de la ubicación del usuario y las rutas generadas, cumplió con el cuarto objetivo del proyecto y mejorando la experiencia del usuario al momento de moverse por el caso de estudio.
5. **Limitaciones identificadas:** Se observaron desafíos significativos relacionados con la inconsistencia en la disponibilidad de datos de posición y las interferencias propias del entorno físico, contribuyendo a el error de la posición en tiempo real obtenida.
6. **Viabilidad de la solución propuesta:** A pesar de las limitaciones mencionadas, el sistema desarrollado demuestra ser una solución viable para la navegación en interiores, ofreciendo mejoras sobre los datos de posicionamiento sin procesar y cumpliendo con los objetivos principales del proyecto.

Estos logros permiten una mejora de la experiencia del usuario en espacios cerrados, ofreciendo la oportunidad para aumentar la eficiencia operativa y la satisfacción del cliente. Sin embargo, las limitaciones identificadas presentan oportunidades para futuras investigaciones y mejoras del sistema.

5.1. Trabajo futuro

Basándonos en los resultados y limitaciones, se proponen las siguientes líneas de investigación futura:

- Explorar técnicas avanzadas de fusión de sensores, como la integración de datos de sensores inerciales (IMU) con Wi-Fi, para mejorar la precisión del posicionamiento y reducir el RMSE y MAE actuales.
- Investigar métodos para manejar de manera más robusta los huecos en los datos de posicionamiento, mediante la implementación de algoritmos de inteligencia artificial o la integración de datos de múltiples fuentes.
- Realizar pruebas extensivas en diversos tipos de espacios interiores, como centros comerciales de diferentes tamaños y niveles, hospitales y universidades, para evaluar la generalización del sistema y su adaptabilidad a diferentes contextos.
- Mejorar la interfaz de usuario de la aplicación móvil, incorporando feedback de usuarios reales y realizando estudios de usabilidad para optimizar la experiencia del usuario final.
- Investigar la posibilidad de implementar aprendizaje automático para mejorar la precisión de la localización y la generación de rutas con el tiempo, basándose en los patrones de movimiento de los usuarios.

ANEXOS

System Technology	Cost (Antenna) (May 2020)	Cost (Receiver) (May 2020)	Infrastructure Impact	Privacy/Security	Power Consumption	Coverage	Scalability	Accuracy	Impact of metal interference	Line of Sight Needed	Signal Type	Limitation
GNSS	N/A	£10-£50	None	High	> 150 mW-1.5 W	Global	High	3 m-15 m+	Moderate to High	Yes	Radio Frequency	Multipath impacts and signal attenuation
Inertial System (MEMS)	≈£15-£30+		None	Med - High	Very Low	10 m ² -2500 m ² +	High	0.8-2% of path travelled Drift can impact error by >0.5 m/s ²	None	No	Environmental /Physical Phenomena	Relative location only
Wi-Fi	≈£5-£10	≈£30-£150+	High unless already existing	Med - High	> 500 mW	10 m ² -2500 m ² +	High	0.4 m-5 m	Moderate	No	Radio Frequency (2.4 GHz or 5 GHz)	Sensor bias and noise Ambient Interference Low accuracy Expensive if not installed
Bluetooth LE	≈£5-£30		Moderate	Med - High	1 mW -50 mW	10 m ² -1000 m ² +	Moderate	0.9 -2 m	High	No	Radio Frequency (2.4 GHz)	Impacted by ambient interference
WSN (Zigbee)	≈£5-£30+		Moderate	Low-Med	17.68 mW	10 m ² -100 m ²	High	0.8-5 m	Moderate	No	Radio Frequency (2.4 GHz)	Low Accuracy Short range and high latency 802.15.4 protocol Less used
UHF RFID (Passive)	≈£100-£250 + £50-£100 per antenna	£0.01-£0.03	Low	Low - Med	> 300 mW	10 m ² -1000 m ² +	Moderate	1 m-5 m	High	No	Radio Frequency (860 MHz to 960 MHz)	High Antenna and Reader cost Severe impact of metal
UHF RFID (Active)	£250+ + £50-£100 per antenna	£50-£100+	Moderate	Med	> 250 mW for Reader	10 m ² -1000 m ² +	Moderate	0.1 m-1 m	Moderate	No	Radio Frequency (860 MHz to 960 MHz)	Metal interference High unit cost Frequent maintenance
Ultra Wideband	≈£10-£200		Low-Moderate	Med-High	> 1 W	10 m ² -400 m ²	Low	0.3 m-0.5 cm	Moderate	No	Radio Frequency (3.1 GHz to 10.6 GHz, 500 MHz bands)	High power consumption High infrastructure impact
Acoustic (Ultrasound)	≈£2-£20	≈£10-£20	Low-Moderate	High	> 300 mW	10 m ² -2500 m ² +	High	0.1 m-0.6 m	None	Yes	Ultrasonic (20 kHz)	Line of sight Ambient sound interference
Vision	≈£10-£250		High unless already existing	Low - High	> 1 W	≈6 m (Resolution and LoS dependant)	Low	1 mm -0.3 m	None	Yes	Light	Requires LoS conditions Impacted by lighting conditions

Figura 37: Resumen de enfoques tecnológicos y precisiones correspondientes para la investigación y tecnologías IPS disponibles comercialmente.

Fuente: [Hayward et al., 2022]

REFERENCIAS BIBLIOGRÁFICAS

- [ind, 2024] (2024). Indooratlas - indoor positioning system. <https://www.indooratlas.com/>. Accedido: 10 de julio de 2024.
- [ipi, 2024] (2024). Ipin - indoor positioning and indoor navigation conference. <https://ipin-conference.org/>. Accedido: 10 de julio de 2024.
- [Alarifi et al., 2016] Alarifi, A., Al-Salman, A., Alsaleh, M., Alnafessah, A., Al-Hadhrami, S., Al-Ammar, M. A., y Al-Khalifa, H. S. (2016). Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 16(5).
- [Almadani et al., 2019] Almadani, Y., Ijaz, M., Joseph, W., Bastiaens, S., Rajbhandari, S., Adebisi, B., y Plets, D. (2019). A novel 3d visible light positioning method using received signal strength for industrial applications. *Electronics*, 8(11).
- [Azmi et al., 2018] Azmi, N. A., Samsul, S., Yamada, Y., Mohd Yakub, M. F., Mohd Ismail, M. I., y Dziyauddin, R. A. (2018). A survey of localization using rssi and tdoa techniques in wireless sensor network: System architecture. En *2018 2nd International Conference on Telematics and Future Generation Networks (TAFGEN)*, pp. 131–136.
- [Basiri et al., 2017] Basiri, A., Lohan, E. S., Moore, T., Winstanley, A., Peltola, P., Hill, C., Pouria, A., y Silva, P. (2017). Indoor location based services challenges, requirements and usability of current solutions. *Computer Science Review*, 24.
- [Belhadi et al., 2014] Belhadi, Z., Fergani, L., Fergani, B., y Laheurte, J.-M. (2014). Rfid tag indoor localization by fingerprinting methods. volumen 0.
- [Bradski y Kaehler, 2008] Bradski, G. y Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.
- [Bulitko y Lee, 2006] Bulitko, V. y Lee, G. (2006). Learning in real-time search: A unifying framework. *Journal of Artificial Intelligence Research*, 25:119–157.
- [Busu y Busu, 2021] Busu, C. y Busu, M. (2021). An application of the kalman filter recursive algorithm to estimate the gaussian errors by minimizing the symmetric loss function. *Symmetry*, 13(2).
- [Choset et al., 2005] Choset, H. M., Hutchinson, S., Lynch, K. M., Kantor, G., Burgard, W., Kavraki, L. E., y Thrun, S. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press.
- [Edelsbrunner et al., 1990] Edelsbrunner, H., Tan, T. S., y Waupotitsch, R. (1990). An $O(n^2 \log n)$ time algorithm for the minmax angle triangulation. En *Proceedings of the Sixth Annual Symposium on Computational Geometry, SCG '90*, p. 44–52, New York, NY, USA. Association for Computing Machinery.

- [Farjad *et al.*, 2019] Farjad, B., Gupta, A., Sartipizadeh, H., y Cannon, A. (2019). A novel approach for selecting extreme climate change scenarios for climate change impact studies. *Science of The Total Environment*, 678.
- [Ferguson *et al.*, 2005] Ferguson, D., Likhachev, M., y Stentz, A. (2005). A guide to heuristic-based path planning. En *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*.
- [Ghilani, 2017] Ghilani, C. D. (2017). *Adjustment computations: spatial data analysis*. John Wiley & Sons.
- [Gonzalez y Woods, 2018] Gonzalez, R. C. y Woods, R. E. (2018). *Digital image processing*. Pearson.
- [Google Maps, 2024a] Google Maps (2024a). Indoor maps. Google Maps website. Accessed: 2024-05-26.
- [Google Maps, 2024b] Google Maps (2024b). Madison square garden - google maps. Accedido el 7 de mayo de 2024.
- [GPS.gov, 2024] GPS.gov (2024). Space segment. Último acceso: 11 de mayo de 2024.
- [Gu *et al.*, 2009] Gu, Y., Lo, A., y Niemegeers, I. (2009). A survey of indoor positioning systems for wireless personal networks. *IEEE Communications surveys & tutorials*, 11(1):13–32.
- [Hansel, 2023] Hansel, B. (2023). Navigation fixes and triangulation. <https://www.paddlinglight.com/articles/navigation-fixes-and-triangulation/>. Accedido el [fecha de acceso].
- [Hayward *et al.*, 2022] Hayward, S., van Lopik, K., Hinde, C., y West, A. (2022). A survey of indoor location technologies, techniques and applications in industry. *Internet of Things*, 20:100608.
- [IEEE Standards Association, 2024] IEEE Standards Association (2024). The Evolution of Wi-Fi Technology and Standards. Último acceso: 12 de mayo de 2024.
- [Inc., 2019] Inc., A. (2019). Introducing indoor maps. Apple Developer website. Accessed: 2024-05-26.
- [Inc., 2024] Inc., A. (2024). Apple indoor maps program. Apple Developer website. Accessed: 2024-05-26.
- [IndoorAtlas, 2023] IndoorAtlas (2023). Indooratlas positioning accuracy. Technical report, IndoorAtlas. Informe técnico interno.
- [Isaac, 2011] Isaac, M. (2011). New version of google maps brings indoor floor plans to your phone. Wired website. Accessed: 2024-05-26.

- [Jaillet y Wagner, 2008] Jaillet, P. y Wagner, M. (2008). *Online Vehicle Routing Problems: A Survey*, volumen 43, pp. 221–237.
- [Kokash, 2005] Kokash, N. (2005). An introduction to heuristic algorithms. *Department of Informatics and Telecommunications*, pp. 1–8.
- [Korf, 1990] Korf, R. E. (1990). Real-time heuristic search. *Artificial Intelligence*, 42(2):189–211.
- [Kos et al., 2010] Kos, T., Markežic, I., y Pokrajcic, J. (2010). Effects of multipath reception on gps positioning performance. En *Proceedings ELMAR-2010*, pp. 399–402.
- [Lang, 1969] Lang, T. (1969). The generalization of geographical maps by means of electronic data processing. *Nachrichten aus dem Karten-und Vermessungswesen*, 2(5):47–58.
- [Latombe, 1991] Latombe, J.-C. (1991). *Robot motion planning*. Springer Science & Business Media.
- [Liu et al., 2007] Liu, H., Darabi, H., Banerjee, P., y Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080.
- [Madkour et al., 2017] Madkour, A., Aref, W. G., Rehman, F. U., Rahman, M. A., y Basalamah, S. (2017). A survey of shortest-path algorithms.
- [Meraki, 2024] Meraki, C. (2024). Location analytics: Location and privacy. Cisco Meraki Documentation. Accessed: 2024-05-26.
- [Mordor Intelligence, 2024] Mordor Intelligence (2024). Indoor location market - growth, trends, covid-19 impact, and forecasts (2024 - 2029). Accessed on [inserta aquí la fecha de acceso].
- [Oliphant, 2006] Oliphant, T. E. (2006). *A guide to NumPy*, volumen 1. Trelgol Publishing USA.
- [Oliveira et al., 2019] Oliveira, L., Schneider, D., De Souza, J., y Shen, W. (2019). Mobile device detection through wifi probe request analysis. *IEEE Access*, 7:98579–98588.
- [Pinandito et al., 2023] Pinandito, A., Kharisma, A. P., y Akbar, M. A. (2023). Improving route-finding performance of dijkstra algorithm and maintaining path visual cue using douglas-peucker algorithm. En *Proceedings of the 8th International Conference on Sustainable Information Engineering and Technology, SIET '23*, p. 401–408, New York, NY, USA. Association for Computing Machinery.
- [Potortì et al., 2023] Potortì, Francesco and Barsocchi, Paolo and Girolami, Michele and others (2023). Evaluating indoor localization solutions in large environments through competitive benchmarking: The ipin 2023 competition. En *2023 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8. IEEE.

- [Rafiq et al., 2020] Rafiq, A., Kadir, T. A. A., e Ihsan, S. N. (2020). Pathfinding algorithms in game development. *IOP Conference Series: Materials Science and Engineering*, 769(1):012021.
- [Rahman, 2012] Rahman, M. Z. (2012). Beyond trilateration: Gps positioning geometry and analytical accuracy. En Jin, S., editor, *Global Navigation Satellite Systems Signal, Theory and Applications*, pp. 241–256. InTech, Rijeka, Croatia.
- [Saalfeld, 1999] Saalfeld, A. (1999). Topologically consistent line simplification with the douglas-peucker algorithm. *Cartography and Geographic Information Science*, 26(1):7–18.
- [Statista, 2024] Statista (2024). Sistema operativo móvil con la mayor cuota de mercado por país. Accedido: 31 de mayo de 2024.
- [Stentz, 1994] Stentz, A. T. (1994). Optimal and efficient path planning for partially-known environments. En *Proceedings of (ICRA) International Conference on Robotics and Automation*, volumen 4, pp. 3310 – 3317.
- [Szeliski, 2010] Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.
- [U.S. Government, 2023] U.S. Government (2023). Gps accuracy. Accessed: 2023-05-11.
- [Vanhoef et al., 2016] Vanhoef, M., Matte, C., Cunche, M., Cardoso, L. S., y Piessens, F. (2016). Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms. En *Proceedings of the 11th ACM on Asia conference on computer and communications security*, pp. 413–424.
- [Visvalingam y Whyatt, 1993] Visvalingam, M. y Whyatt, J. D. (1993). Line generalisation by repeated elimination of points. *The Cartographic Journal*, 30(1):46–51.
- [Yang et al., 2018] Yang, J., Li, Y., y Cheng, W. (2018). An improved geometric algorithm for indoor localization. *International Journal of Distributed Sensor Networks*, 14:155014771876737.
- [Yassin et al., 2017] Yassin, A., Nasser, Y., Awad, M., Al-Dubai, A., Liu, R., Yuen, C., Raulefs, R., y Aboutanios, E. (2017). Recent advances in indoor localization: A survey on theoretical approaches and applications. *IEEE Communications Surveys Tutorials*, 19(2):1327–1346.
- [Zafari et al., 2019] Zafari, F., Gkelias, A., y Leung, K. K. (2019). A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials*, 21(3):2568–2599.
- [Zhuang et al., 2016] Zhuang, Y., Yang, J., Li, Y., Qi, L., y El-Sheimy, N. (2016). Smartphone-based indoor localization with bluetooth low energy beacons. *Sensors*, 16(5).