Universidad Zaragoza

Trabajo Fin de Grado

Implementación de algoritmos de partición en celdas para la planificación de trayectorias de robots móviles

Cell decomposition algorithms for path planning of mobile robots

Autor

Pedro Cabello Díaz

Directores

Eduardo Montijano Muñoz

Cristian Florentín Mahulea

ESCUELA DE INGENIERÍA Y ARQUITECTURA
Departamento de Informática e Ingeniería de Sistemas

2021

Resumen

La robótica es una tecnología en auge y los robots paulatinamente están pasando a formar parte de nuestra vida cotidiana, ya que actualmente permiten la realización de acciones que habían sido usualmente consideradas como complicadas o pesadas sin apenas problema. Una de las modalidades de robots que ha experimentado un mayor avance es la robótica móvil. Estos robots son capaces de desplazarse a lo largo de una ruta, habitualmente construida con la ayuda de un grafo obtenido a partir de un mapa. Para obtener esta información una vez que el mapa es conocido se requiere de una descomposición en celdas a través de las cuales posteriormente habrá de desplazarse el robot.

En este trabajo se presentan métodos de segmentación de mapas en celdas para la posterior planificación de trayectorias de forma que el robot alcance un lugar designado como objetivo partiendo de un origen conocido. También se muestra el procedimiento seguido para el desarrollo íntegro de software que contiene a dos de los algoritmos de partición de imágenes, escritos en el lenguaje de programación orientado a objetos C++. Así pueden estos programas ser empleados como un nodo dentro de ROS (*Robot Operating System*), la principal librería de software en el ámbito de la robótica móvil. Estos algoritmos generan en un fichero de texto una lista de celdas caracterizadas por su información más destacada: etiquetado, ocupación, sus dimensiones y las relaciones que guardan con sus celdas adyacentes.

Para la generación de un mapa topológico modelado como un grafo, se parte de una imagen del plano en el que ha de moverse el robot. Tras desarrollar los algoritmos de partición, se realiza un análisis y comparación de diferentes métodos a través de métricas, para así encontrar el algoritmo de descomposición en celdas más preciso y con un menor coste computacional. Tras evaluar cual de ellos resulta más atractivo a la hora de generar un mapa para un robot según la aplicación deseada se concluye con un comentario crítico del trabajo realizado.

Abstract

Robotics is currently undergoing a rapid growth and therefore robots are gradually becoming part of our everyday life. The main reason for this is that they allow us to carry out actions that until then had usually been considered very complicated or tiresome with hardly any problems. One of the main types of robots are mobile robots, which have among their main abilities the capability to move along a route. To trace this path, they are usually based on a graph obtained from a map, which, once it is known, requires a partition into cells in which the robot will move.

This paper presents the main methods of segmenting maps into cells for the subsequent planning of trajectories for a robot to reach a designated target location. It will also show the procedure followed for the development of software containing such partitioning algorithms, written in the object-oriented programming language C++, so that it can be used as a node within the main robotics framework, ROS.These algorithms provide a list of cells characterised by their salient information, such as their label, dimensions and adjacent cells, all in a text file.

The generation of a graph containing the free cells and obstacles, i.e. a topological map, is based on an image of the plane in which the robot will have to move. After developing the partitioning algorithms, an analysis and comparison of the different methods will be carried out using various metrics in order to find the most accurate method of splitting the image and with a lower computational cost.In this way, it will be evaluated which of them is more attractive when it comes to generating a map for a robot according to the desired application. Finally, after an overview about the work has been done there will be a crytical review about the contents of this paper.

Agradecimientos

Me gustaría agradecer este trabajo a todas las personas que me han acompañado durante la realización del mismo, especialmente a mi familia.

Índice general

Índice de figuras

Índice de tablas

Índice de fragmentos de código

Capítulo 1

Introducción

A lo largo de este capítulo se describe la temática del trabajo, así como el ámbito y motivos del mismo. Además se tratan los objetivos y el desarrollo realizado para alcanzarlos.

1.1. Motivación y contexto

La robótica es un campo de investigación que cada vez resulta una parte más importante de nuestras vidas, siendo empujada por un avance continuo de los medios tecnológicos. Hoy en día entre sus objetivos está el de dar a los robots una mayor autonomía, alcanzando una de sus mejores representaciones en la robótica móvil.

Los robots móviles son aquellos que pueden desplazarse en el medio en el que se encuentran. Ejemplos de ello son los robots de limpieza, usados de manera cotidiana en muchos hogares, robots de transporte de materiales en fábricas, o incluso los cada vez más autónomos y seguros vehículos eléctricos, que en poco tiempo van a formar parte de nuestra vida cotidiana [1].

La tarea de transformar imágenes que contengan mapas en los caminos en los que el robot pueda moverse es un paso fundamental de una de las vías de generar rutas en la robótica móvil [2]. A través de la descomposición de los mapas en celdas es posible conseguir una mayor modularidad y por lo tanto una mejor eficiencia para la planificación de trayectorias de un robot.

El mapeo es una rama aplicada de la cartografía y de la visión artificial [3] mediante la que se estudia y se construye un mapa. Para realizar una descomposición en celdas se parte de un mapa conocido inicialmente. De esta manera el robot se localiza en el mapa y posteriormente puede trazar las rutas para alcanzar un destino. Gracias al mapeo y la generación de rutas la respuesta del robot no es solo instintiva: no responde únicamente a los impulsos que recibe, sino que las rutas se trazan previamente. La extensión de ello permite dar un enfoque más humano a la robótica mediante la creación de mapas cognitivos, en los que se emplea el reconocimiento de patrones y lo esperado de ellos para que un robot alcance su destino de una forma óptima [4].

Hoy en día es necesario que los robots móviles tengan un mapa topológico en el que todos sus movimientos puedan ser planificados según los algoritmos con los que han sido programados. Para ello es útil que este mapa esté subdividido en celdas, que puedan ser fácilmente distinguibles entre obstáculos y vías de avance para el robot. Estas celdas han de poderse ubicar sin problema en el mapa y además, es conveniente que muestren el tipo de relación que guardan cada una de ellas con sus celdas adyacentes a fin de simplificar la posterior labor de planificación de rutas.

Desde muchas empresas, centros de investigación, departamentos universitarios y otras instituciones se lleva desde los comienzos de la robótica tratando de encontrar una metodología de descomposición de mapas óptima para optimizar la labor de generación de estructuras de datos a partir de los que generar rutas [2]. Este método ha de aunar la precisión con la sencillez, que conlleve ligereza a la hora de manejar la carga computacional necesaria para su realización.

1.2. Objetivos y alcance del trabajo

El objetivo principal de este trabajo es la implementación de algoritmos de descomposición en celdas un mapa y la generación de grafos, con los que puedan trazarse las rutas adecuadas para un robot móvil.

Con objeto de lograr un proyecto consistente este objetivo principal se ha descompuesto en los siguientes hitos:

- Transformación de cualquiera de los principales formatos de imagen en el formato PGM (*Portable Gray Map*) para generar la partición.
- Descomposición de la imagen en celdas relacionadas entre si.
- Formación de un mapa del cual se puede extraer la información más importante.

- Generación de un documento que contenga los resultados de la partición (grafo) en un formato fácilmente accesible.

Este trabajo ha supuesto para el autor una toma de contacto y aprendizaje del mundo de la programación informática y el funcionamiento de los robots, que ha resultado muy interesante para su formación académica. El trabajo se ha desarrollado en base a los principios del software libre, de forma que resulte accesible y escalable, manteniendo su espíritu didáctico.

Como ya se ha dicho, este proyecto representa un estudio e implementación completa de determinados algoritmos de descomposición en celdas. La labor realizada se ha desarrollado en las siguientes fases:

- Familiarización y estudio del contenido del trabajo, abordando específicamente:

  Métodos de descomposición de un mapa en celdas para la posterior generación de rutas en el ámbito la robótica móvil.

  El lenguaje de programación C++, en el que se han escrito todos los algoritmos desarrollados.

- Implementación de los algoritmos que han sido elaborados en su totalidad por el autor de este proyecto. Posteriormente se ha verificado su funcionamiento y la posibilidad de ser puestos en práctica en el entorno de software de robótica ROS para la generación de rutas.

- Análisis comparativo de los algoritmos implementados a través del diseño de experimentos que permiten categorizar y extraer las características de lo programado. Se ha tratado de dar cabida no solo al análisis del funcionamiento de los métodos de descomposición en celdas generados, sino también de posteriores implementaciones de otros métodos de descomposición.

Todo el software utilizado en este trabajo es software libre, siendo esta una de las premisas en las cuales se basan los desarrollos. De esta forma se logra una interoperabilidad de lo generado y se evitan los problemas derivados de la propiedad intelectual del software. Además, se ofrece el acceso libre al software desarrollado a través de GitHub, la mayor plataforma de creación de proyectos colaborativos en la actualidad [5]. El link para acceder al repositorio es el siguiente:

https://github.com/740045/TFG.git

Se consideran alcanzados los objetivos del trabajo en cuanto al desarrollo realizado en la generación del código fuente que contiene a los algoritmos de descomposición en celdas, así como plataformas de creación aleatoria de mapas para su evaluación. Para completar el trabajo se han empleado numerosas plataformas y herramientas al alcance de cualquier persona con conexión a Internet. Para ilustrar que partes del trabajo son de elaboración propia y cuales se han tomado de repositorios ya existentes se puede consultar la Figura 2.1 contenida al final del Capítulo 2.

1.3. Estructura de la memoria

En este Capítulo 1 se expone una breve introducción del trabajo y se describen los objetivos del mismo, ilustrando la problemática que conlleva la generación de un grafo que servirá para trazar rutas a partir de la descomposición en celdas de un mapa. Para más detalles sobre las tecnologías del entorno de la robótica móvil en el que se desarrolla el proyecto, así como para aclarar otros conceptos mencionados en la memoria se puede consultar el Anexo A.1 (Estado actual de las tecnologías relacionadas).

Los principales métodos para abordar las dificultades que conciernen a la obra se exponen en el Capítulo 2 (Metodología y herramientas empleadas). Para ello es necesario hacer un interludio en el Capítulo 3 (Algoritmos de descomposición en celdas) donde se describen diferentes métodos de descomposición de un mapa en celdas. Entre los algoritmos de descomposición de celdas tratados se seleccionan dos de ellos para ser implementados en C++ tal y como queda descrito en esa sección. Para este proceso se han utilizando varias herramientas informáticas explicadas en el Anexo A.2 (Herramientas informáticas empleadas). También se hace mención de varios fragmentos de código que se han considerado fundamentales, que son mostrados en el Anexo A.3 (Fragmentos de código mencionados).

El método escogido para comunicar los resultados obtenidos de las particiones, así como ejemplos de los diferentes mapas empleados para poner a prueba los métodos implementados se muestran en los Anexos A.4 y A.5 respectivamente (Ejemplos de los mapas generados y Contenido del grafo respectivamente).

Los algoritmos que han sido desarrollados por el autor se analizan y comparan en el Capítulo 4 (Estudio comparativo de los algoritmos implementados). Los programas que se han escrito para la comparativa se comentan brevemente en el Anexo A.6 (Algoritmos de comparación entre los diferentes métodos implementados). Se muestran los resultados obtenidos en el Anexo A.7 (Resultados de las comparaciones realizadas).

Para concluir la memoria se van a extraer los conceptos más destacados en el Capítulo 5 (Conclusiones), en el cual también se realiza una visión de conjunto de las vías de desarrollo

plausibles para la continuación del proyecto, al igual que las incidencias encontradas y las vías empleadas para solucionarlas.

En la Figura 1.1 se muestra un cronograma de los procesos globales y subprocesos de los que ha constado este proyecto. De esta forma se puede observar como la evolución del conocimiento sobre la materia a tratar ha permitido sostener el desarrollo y posterior análisis de los algortimos implementados.

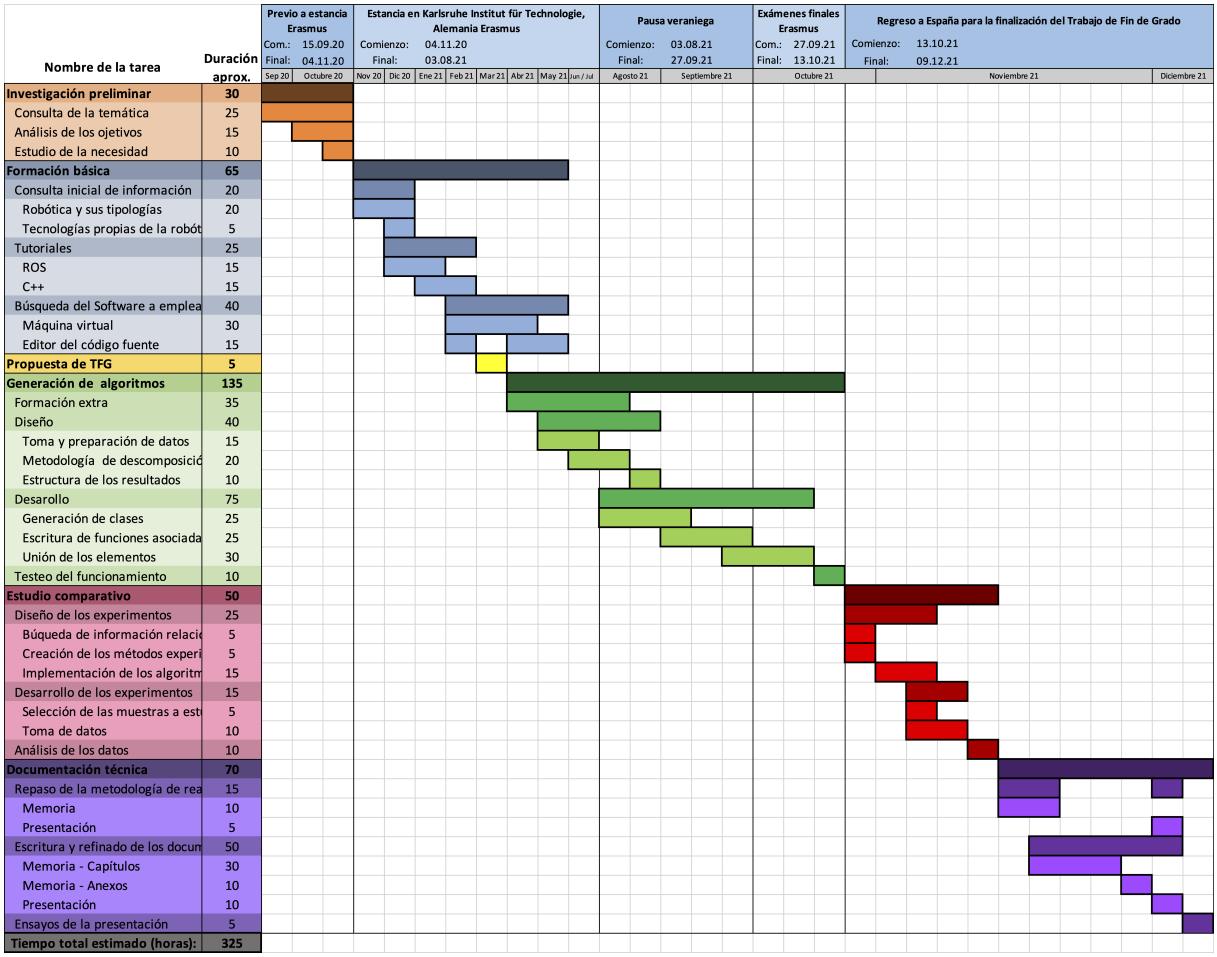


Figura 1.1: Cronograma de las diferentes tareas del desarrollo de este trabajo.

Capítulo 2

Metodología y herramientas empleadas

La pregunta que contiene el enfoque del trabajo es de qué manera se puede obtener una división de un mapa contenido en una imagen para que el robot pueda generar una ruta a través de él. Para resolver dicha cuestión se sigue una metodología ilustrada a través del origen de los contenidos utilizados: Los cimientos del trabajo desarrollado se han construido a partir de fuentes de información externas relacionadas con el mundo de la robótica. Para ello durante los últimos meses de 2020 mediante el desarrollo de tutoriales recomendados por los directores del trabajo se establecen los conocimientos imprescindibles para el trabajo posterior.

A la hora de describir las características y herramientas para resolver la problemática tratada en el trabajo se ha contado con las aportaciones fruto de conversaciones con estudiantes del *Karlsruhe Institut für Technologie*, universidad donde se ha realizado una estancia formativa durante el desarrollo de este trabajo. Durante el periodo entre los semestre de invierno y verano se realiza en la biblioteca de la mencionada universidad una lectura de fragmentos de libros mencionados en la bibliografía del trabajo.

Destacan también los aportes de la documentación obtenida en internet, así como de otras lecturas, como la obra de E.S. Raymond de 1999 *"The Cathedral and the Bazaar"* [6]. Gracias a este libro se ha reforzado la idea de tomar como uno de los pilares del trabajo la disponibilidad del mismo al público general a través del repositorio de la Universidad de Zaragoza denominado Zaguan en el URL www.zaguan.unizar.es, así como del empleo de métodos colaborativos para resolver algunos problemas encontrados durante la realización del TFG.

Una vez los datos fueron considerados como suficientes para poder implementar los algoritmos de resolución del problema, se cambió hacia un modus-operandi basado en métodos de obtener la solución del problema desarrollados propiamente por el autor. Pa-

ra ello se trató de desarrollar todo el código necesario para la creación de los programas necesarios.

El procedimiento escogido prosperó frente al uso de código ya escrito previamente por otros creadores de contenido debido tanto a cuestiones didácticas como por la flexibilidad que ofrece: las soluciones generadas se adaptadarían perfectamente a las delimitaciones acordadas con los directores del TFG a comienzos de este. Además se ha podido completar uno de los objetivos personales que impuestos a escoger este trabajo: ampliar la formación en las ciencias de la computación.

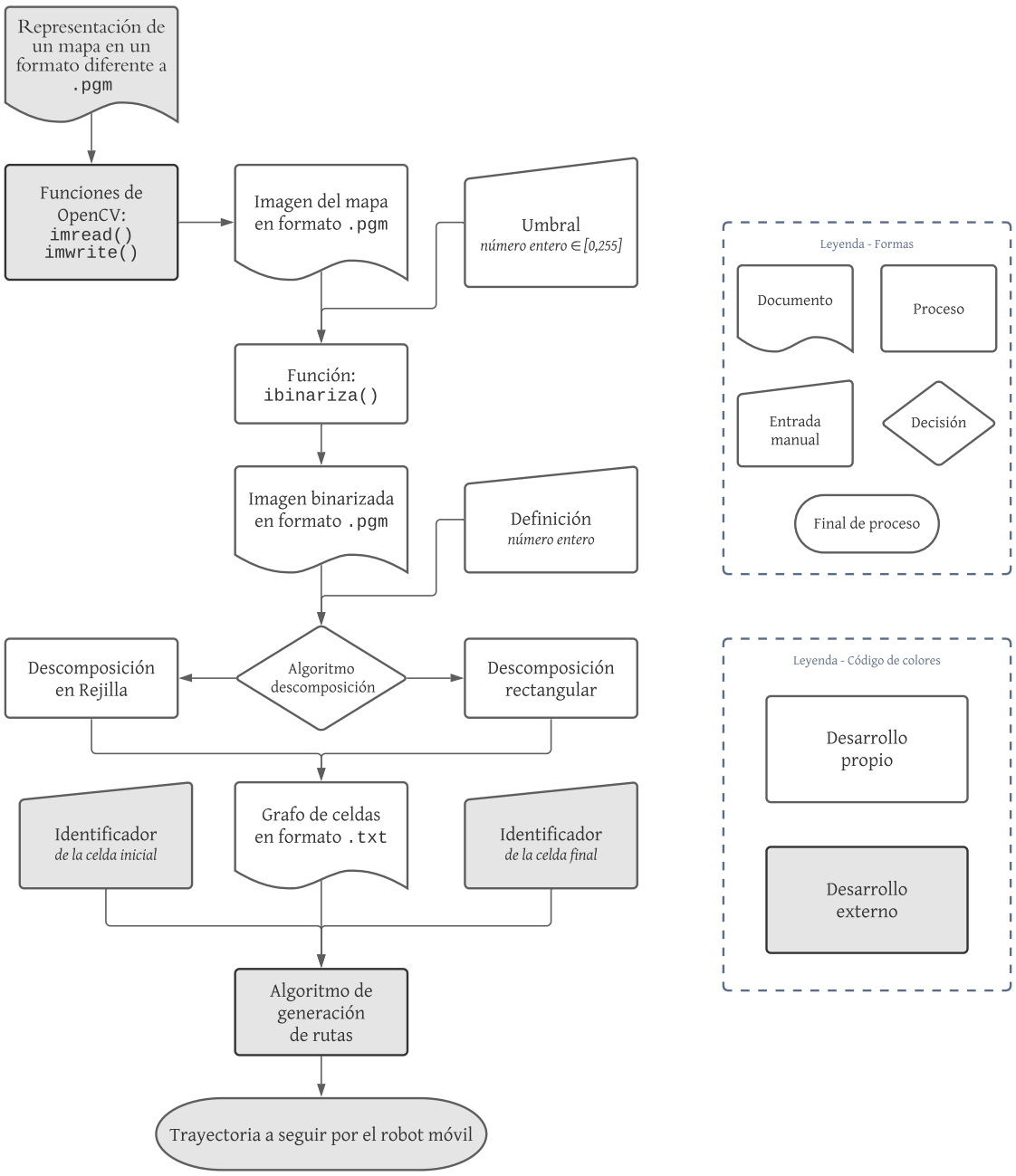
La Figura 2.1 muestra un diagrama de flujo que representa el procedimiento de generación de rutas partiendo de una imagen del mapa a través del cual ha de circular el robot. Se puede ver con fondo blanco todo lo generado por el autor de este trabajo, mientras que con un fondo grisáceo se muestran las aportaciones externas empleadas.

Se van a describir los elementos que aparecen en el diagrama:

- *Documento*: Principal contenedor de la información necesaria para la generación de rutas.
- *Proceso*: Actividades y funciones que permiten transformar unos Documentos en otros diferentes, basándose en como son programados y en las Entradas manuales.
- *Entrada manual*: Aportación eterna, puede ser modificada en el código implementado.
- *Decisión*: Situación en la que existe la posibilidad de decidir por que camino se ha de generar la solución deseada.
- *Final de proceso*: Resultado deseado del proceso global y finalización de las tareas necesarias para obtenerlo.

Gracias a la librería de software ROS (Anexo A.2.3) el procedimiento de generación de rutas para robots móviles puede ser automatizado. Para ello cada uno de los *procesos* son implementados como nodos, tomando los *documentos* como mensajes estandarizados empleados para la comunicación entre ellos. Tanto el resultado de la *decisión* como las *entradas manuales* se han de establecer antes de hacer correr el programa. Otra opción para ellos sería que dependieran de la idoneidad para el entorno en el que se haya de mover, o de las características de este y del robot.

Figura 2.1: Diagrama de flujo de la generación de una ruta para un robot móvil a partir de un mapa ya conocido.



Capítulo 3

Algoritmos de descomposición en celdas

"Divide y vencerás" es un refrán empleado en la lengua española para hacer referencia el hecho de que al dividir un problema de grandes dimensiones en partes más pequeñas se puede resolver de una manera ligera y habitualmente de una forma más simple. Asimismo se puede optar por un método de resolución colaborativo al fragmentar la resolución en partes para diferentes personas. De esta forma se genera una solución con más riqueza en puntos de vista, a menudo realizada en menos tiempo y finalmente con una calidad superior.

La descomposición adquiere una importancia fundamental a la hora de resolver problemas del calibre de los que suele enfrentarse la robótica. El que es tratado en este trabajo es el del tratamiento y segmentación de imágenes que representarán los mapas a través de los que se ha de desplazar el robot. Sin esta división del mapa sería imposible informar al robot del significado contenido por él, no pudiendo ser empleado para trazar rutas posteriormente por el robot.

3.1. Fundamentos teóricos

Antes de mostrar los algoritmos de partición de celdas más importantes, es esencial conocer el soporte matemático en el que están basados. Partimos de un espacio bidimensional rectangular $Ev \subset \mathbb{R}^2$. Esta restricción a $2D$ se debe a la viabilidad con las descomposiciones a comentar, así como al hecho de que sea este el ámbito más cotidiano para muchos robots hoy en día. [7] En este espacio bidimensional definimos las coordenadas cartesisanas $x_{min}, x_{max}, y_{min}, y_{max} \in \mathbb{R}$ que forman el subespacio que contendrá al plano: $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$.

Este medio plano contendrá obstáculos son representados por politopos[1], que en las representaciones serán mostrados por un color oscuro. Estas regiones serán denominadas como $O_1, O_2, \ldots, O_n$. Sus vértices se denominarán como $V_S = \{({v_S}^1, \ldots, {v_S}^{|V_S|}\}$, mientras que las caras de los polígonos se denominarán $F_S = \{({f_S}^1, \ldots, {f_S}^{|F_S|}\}$. Se va a emplear como convención la numeración de ellos en sentido horario comenzando desde el 1. Las zonas que no sean consideradas como obstáculos serán las que posteriormente definan las celdas. Estas serán designadas como $c_1, c_2, \ldots, c_n$ en las figuras que van a mostrar las diferentes particiones en celdas del ejemplo de plano contenido en la Figura 3.1. Tanto esta imagen como las posteriores representaciones de las particiones han sido extraídas de artículo *"Path Planning of Cooperative MobileRobots Using Discrete Event Models."*, 2020, escrito por C. Mahulea et al. [8].

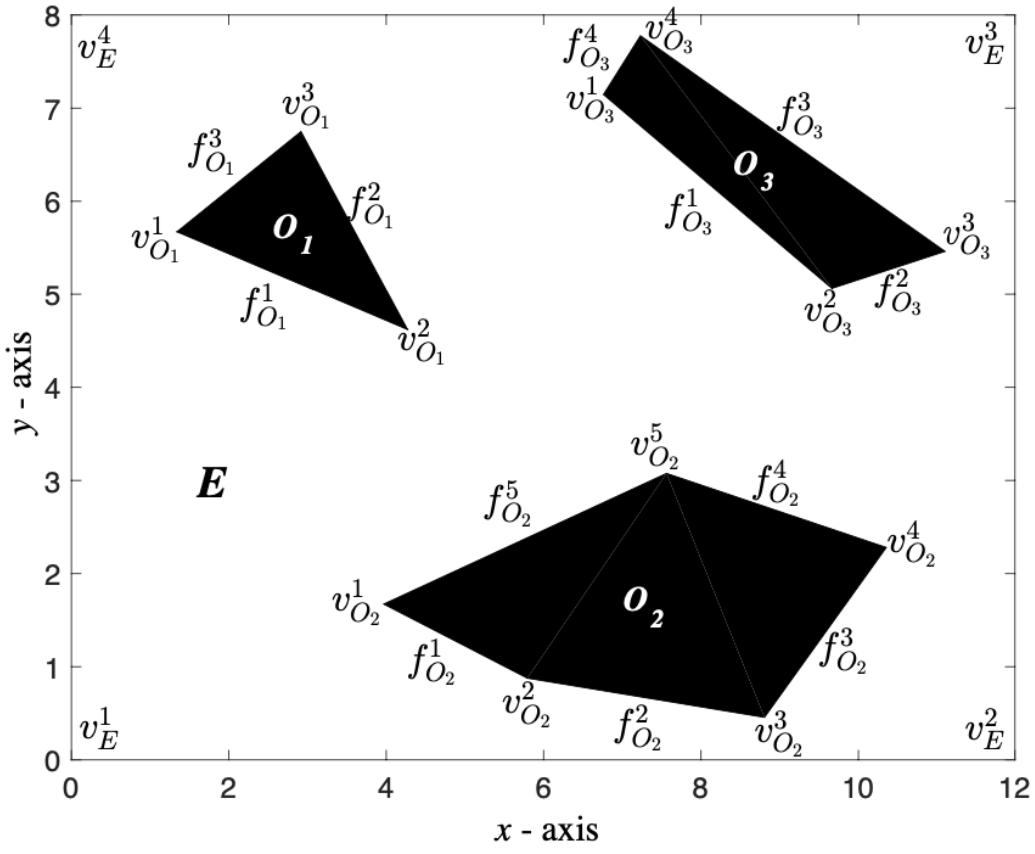


Figura 3.1: Mapa métrico topológico al que se aplicarán las descomposiciones en celdas.

[1]Polígonos con un número indefinido de caras planas.

3.2. Diferentes algoritmos de partición en celdas

En esta sección se van a introducir los métodos más comunes para generar descomposiciones en celdas, explicando en qué está basado cada uno de ellos, así como el procedimiento a seguir para realizarlos. Además, se hará una demostración de su aplicación a la figura que hemos tomado como ejemplo (Fig. 3.1). En todas las descomposiciones se van a mostrar las celdas numeradas así como las relaciones de adyacencia que tienen entre ellas mediante líneas entre sus centroides.

En la Tabla 3.1 se hace una comparativa de las principales características de las diferentes descomposiciones. Dado que en la realidad los obstáculos sean en muchas ocasiones "inflados[2]" en su discretización en píxeles [9], por lo que se ha preferido implementar los métodos de descomposición no exactos. El desarrollo de otros métodos, a pesar de considerarse como interesante, ha quedado fuera debido a las limitaciones temporales del proyecto.

	Retícula	Trapezoidal	Triangular	Politopal	Rectangular
Descomposición exacta	No	Sí	Sí	Sí	No
Celdas adyacentes comparten exactamente una cara	No	No	Sí	Sí	No
Fácilmente generalizable a mayores dimensiones	Sí	No	No	Sí	Sí
Numerosas celdas minúsculas son obtenidas	No	No	No	Sí	No

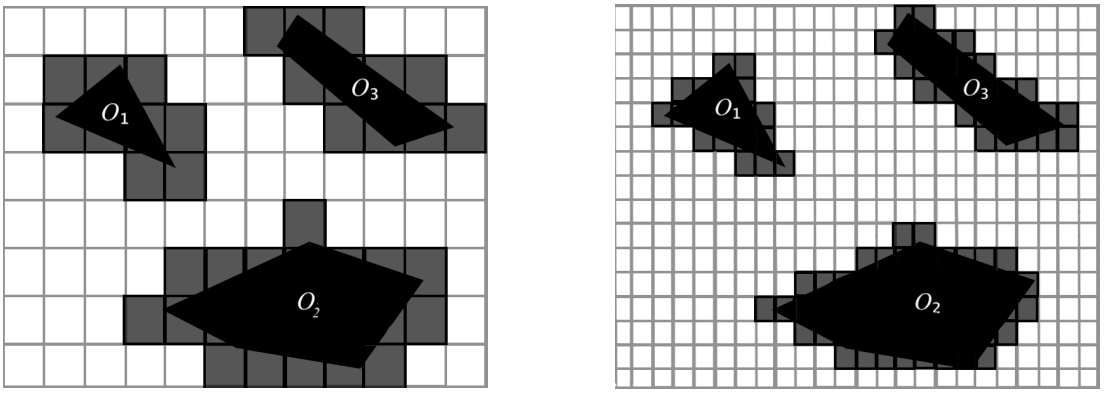
Tabla 3.1: Propiedades (en las filas) frente a los tipos de descomposición (en las columnas).

[2]Quedar representados los obstáculos por dimensiones mayores de las que tienen realmente para así evitar conflictos de colisión del robot móvil con ellos.

3.2.1. Descomposición en retícula

El primer proceso de descomposición en celdas que se explica va a ser una partición simple del mapa denominada en castellano "en retícula", "en malla" o simplemente "en cuadrícula". Para ello la imagen se dividen en cuadrados iguales, siendo estos de un tamaño preestablecido. A continuación se comprobará si las celdas contienen un obstáculo, por lo que pasan a ser denominadas como obstáculo, o bien si están libres de un impedimento para el desplazamiento del robot a través de ellas, cuando serán consideradas como posibles caminos para el robot.

En la aplicación del algoritmo de descomposición en retícula al mapa de ejemplo mostrado en la Figura 3.1 se ha tomado como longitud del borde de cada elemento de la cuadrícula (también denominada a lo largo de la memoria como *definición*) el valor de 1 o 0,5 unidades de las representaciones mostradas en el mapa original y Los resultados se pueden observar en la Figura 3.2.1.



(a) Resolución = 1 unidad (b) Resolución = 0,5 unidades

Figura 3.2: Descomposiciones en retícula.

El valor de la definición tiene una gran influencia en el porcentaje de celdas considerado como obstáculo,que llevándolo al límite al darle una cifra a la definición infinitésimamente pequeño, sería igual a la fracción del área de los obstáculos entre el área total del subespacio Ev.

Esta descomposición, a pesar de perder precisión al existir habitualmente celdas que sean consideradas ocupadas siendo un obstáculo solamente una parte ínfima de ellas, puede ser la base de muchas otras estrategias de partición. Su presumible bajo coste computacional[3] lo hace servir habitualmente como paso intermedio antes de afrontar

[3]Debido a la especificación limitada en del algoritmo

métodos de mapeo basados en reconocimiento de estructuras a través de inteligencia artificial, entre otros [10].

Es una de las particiones implementadas, ya que se ha considerado adecuada en la toma de contacto con la implementación de algoritmos de descomposición en celdas en C++, además de que gracias a pequeño post-procesado de los resultados se puede llegar a resultados mucho más valiosos que los obtenidos en primera instancia. En el trabajo hemos logrado esto gracias a un reconocimiento básico de formas con la función de agrupado de obstáculos..

3.2.2. Descomposición trapezoidal

En ocasiones denominada como "descomposición en celdas verticales", es una descomposición empleada especialmente de forma ilustrativa en problemas de la robótica móvil. Consiste en el trazado de una línea vertical en cada uno de los vértices de los obstáculos. De esta forma se generan celdas irregulares de cuatro vértices, con sus dos lados verticales paralelos. Este método también puede aplicarse con líneas horizontales en vez de verticales.

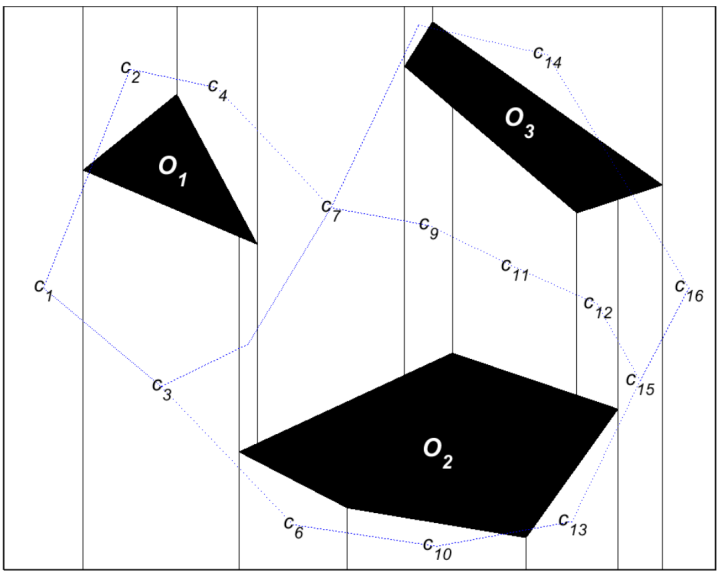


Figura 3.3: Descomposición trapezoidal.

3.2.3. Descomposición triangular

La idea clave del modelado de esta partición reside en el principio de triangulación de Delaunay [11] al emplear como puntos los vértices del mapa y de los obstáculos. Se han de imponer ciertas restricciones a este método, explicado en el Anexo A.8.1. La aplicación del método estándar generaría celdas que contendrían obstáculos y vías libres, generando una duda en la descomposición, que se opondría al objetivo de la partición. Para evitar problemas en el proceso de partición se han de imponer determinadas condiciones, como pueden ser la adición de determinados puntos. En la Figura 3.2.3 se ha optado por exigir que los triángulos se realicen solamente entre los vértices significativos enfrentados (tanto de obstáculos como de los bordes del mapa Ev). Además se ha de evitar que estas triangulaciones interactuen con vértices que no sean los más próximamente enfrentados.

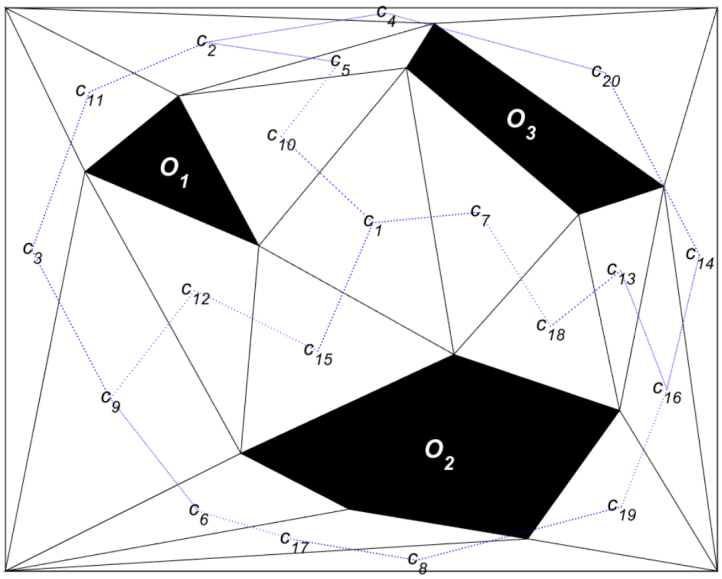


Figura 3.4: Descomposición triangular.

3.2.4. Descomposición politopal

Celdas politopales serán formadas a partir de la prolongación de las aristas de los obstáculos, generándose intersecciones entre ellas. Las celdas son el fruto del encuentro de dichas líneas. Los polígonos formados carecen de ningún tipo de restricción más allá de las caras planas que los delimitan.

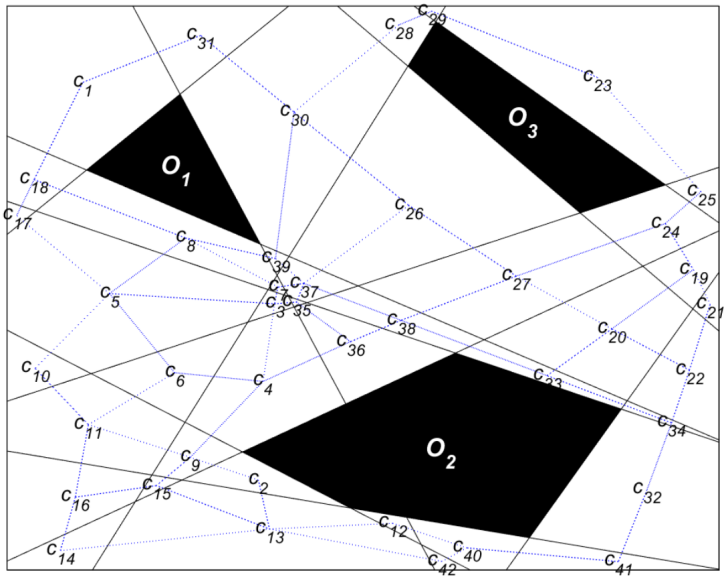


Figura 3.5: Descomposición politopal.

3.2.5. Descomposición rectangular (*Quadtrees*)

Este método de descomposición recursiva denominada como *Quadtrees*, ya que podría ser descrito en un diagrama de árbol, generándose cada vez que se desate un acontecimiento predeterminado cuatro nuevas ramas (Fig. 3.6). Esta metodología guarda similitud con la Descomposición de retícula tratada en el Punto 3.2 debido a que todas las celdas de ambas comparten la misma forma. De todas formas, la Descomposición Rectangular es mucho más eficiente, al ajustar los tamaños de las celdas a su proximidad a un obstáculo. Este procedimiento será explicado en el siguiente párrafo.

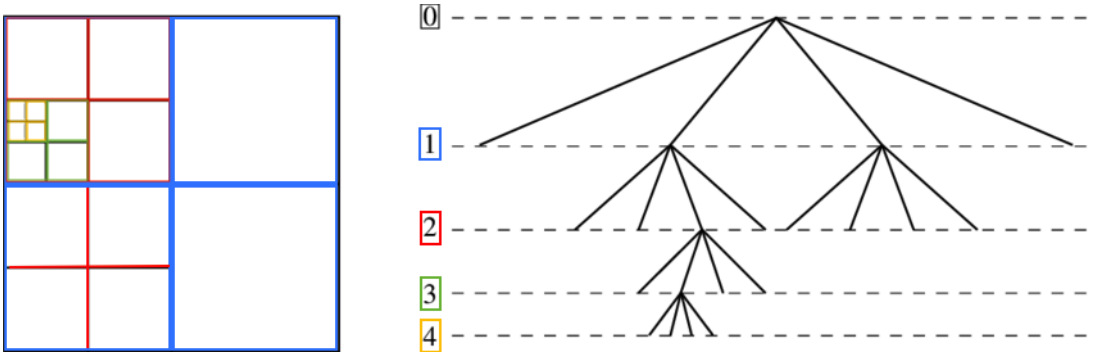


Figura 3.6: Esquema lógico de la descomposición rectangular.

Este método puede ser expandido a tres dimensiones, pasando a ser denominado como *octree* o árbol octal. Además de la aplicación ya descrita en la partición de mapas en el ámbito de la robótica, también es usado en otros campos tan diversos como estructuras de datos [12], desarrollo de juegos [13] o incluso en el modelado del comportamiento de las inundaciones [14].

Para realizar una Descomposición Rectangular hay que partir de la definición de los tres estados posibles de cada una de las celdas: *libre* - cuando ninguno de los píxeles que componen la celda son un obstáculo, por lo que la celda puede ser un camino para la ruta del robot; *ocupada* - todos los píxeles de la celda representan a un mismo obstáculo o bien *mixta*, que será la tipología de las celdas que contengan tanto partes de uno o varios obstáculos como vías libres para que avance el robot. En las celdas mixtas reside la clave de este algoritmo, que aplicará el tratamiento mostrado en la Figura 3.6, que dividirá en cuatro secciones una celda cada vez que encuentre una que sea considerada como mixta.

A primera vista puede parecer que hay celdas pequeñas, pero estas siempre se encontrarán cerca de obstáculos. Esta característica es considerada habitualmente como un aspecto positivo de esta partición, ya que puede conferir al robot la posibilidad de realizar maniobras mucho más precisas cuando se encuentre cerca de impedimentos para su desplazamiento.

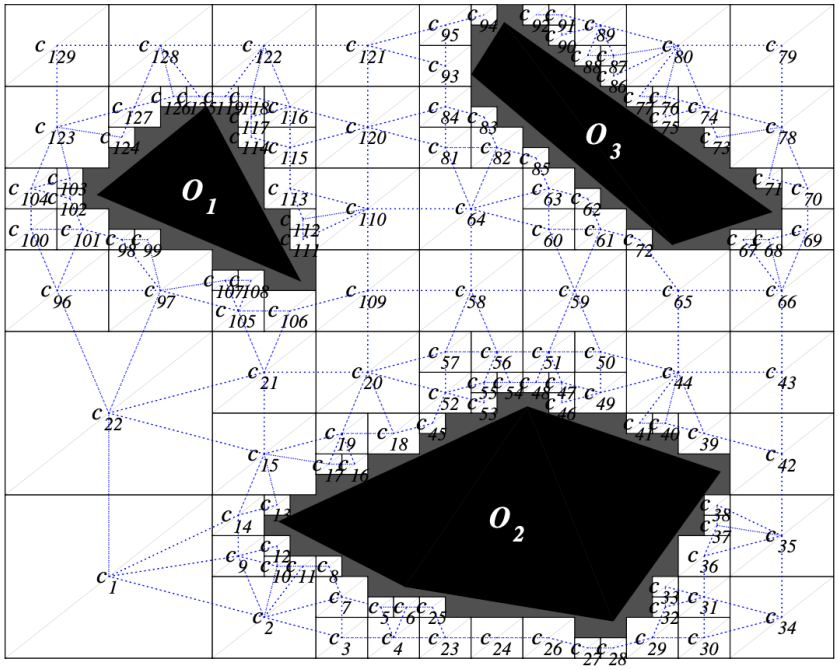


Figura 3.7: Descomposición rectangular.

3.3. Implementación y extensiones

Teniendo en cuenta las características de este proyecto se puede partir tanto de las imágenes proporcionadas por el usuario, así como de las generadas como mapa por un proceso ejecutado en ROS, la mayor plataforma para el desarrollo de software para la robótica móvil (Anexo A.2.3).

Es imprescindible que cualquier formato de imagen que contenga el mapa y que se vaya a ser empleada por los programas sea transformada a formato `.pgm`, ya que así podrá ser procesado por los algoritmos implementados como método para almacenar los datos de la imagen. Para ello se ha utilizado la librería de visión por computador: OpenCV (explicada en mayor profundidad en el Anexo A.2.4).

En esta librería de uso etendido en el campo de la visión por computador existen las funciones de carga de una imagen: `cv::imread()` y de transformación al formato `.pgm`: `cv::imwrite()`. En las siguientes líneas de código aparece un pequeño programa que hace uso de las funciones ya mencionadas.

```
//libreria estandar de C++ ocupada de entrada/salida de datos
#include <iostream>
//declaro las librerias OpenCV
#include <opencv2/opencv.hpp>
#include <opencv2/imgcodecs.hpp>

(...)

// Lectura del archivo de la imagen default.jpg
// ha de encontrarse en la misma carpeta que el programa
//IMREAD_GRAYSCALE genera una lectura en escala de grises
Mat imagen = cv::imread("./default.jpg", IMREAD_GRAYSCALE);

//Escribe la imagen en el formato deseado como mapa.pgm
imwrite("mapa.pgm",imagen);
```

Fragmento de código 3.1: Fragmentos de un programa para transformar imágenes al formato `.pgm` en OpenCV.

A la hora de realizar programas informáticos que modelen estas particiones se ha optado por programar aquellas que no introduzcan diagonales en sus metodologías por su importante carácter didáctico, además de por los errores sistemáticos que se generan al establecer una frecuencia de muestreo con los píxeles que representen a los obstáculos reales, que incrementaría al discretizar formas reales trazando diagonales a partir de ellos. Por ello los algoritmos escogidos para llevar al ámbito informático han sido los no exactos: descomposición en cuadrícula y descomposición rectangular. Pueden encontrarse junto con todo el software desarrollado en el trabajo en el siguiente link:

https://github.com/740045/TFG.git

En este link se puede encontrar el código que ha sido creado para completar los objetivos del trabajo. Se encuentra la carpeta global dividida en varias secciones que tienen cada uno de los proyectos implementados.

3.3.1. Clases destacadas en la implementación

Tal y como es habitual en la metodología estándar en C++, todas las clases comenzarán con mayúscula. Las clases son una de las ventajas que introduce C++ frente a su versión predecesora C debido a que pueden almacenar no solo datos en variables, sino que también tienen de funciones propias. Estas últimas serán explicadas en el punto (3.3.2).

En la escritura del código que representa a las clases se han empleado clases públicas en los programas que he desarrollado, de tal forma que puedan tener acceso a ellas otras

clases. De esta forma se ha conseguido construir los nexos entre ellas necesarios para alcanzar los objetivos propuestos. Esta sección trata las clases básicas más importantes que forman parte de todos los sistemas empleados.

Matrix

El objetivo de este conjunto de datos mostrado en el Fragmento de código A.1 es contener los valores que permiten representar perfectamente la imagen contenida en el documento `mapa.pgm`. De esta forma será más fácil para otras funciones poder acceder a los datos, además de contener determinadas funciones que agilizarán posteriores labores de división en celdas o de análisis.

Celda

Esta entidad comprenderá la información que contiene cada celda, así como alguna función para ayudar a completarlos. La implementación de está función puede verse en el fragmento de código A.2 La función de las celdas es poder contener de las características que tienen para posteriormente poder ser almacenadas y a través de las cuáles podrá circular el robot, o bien evitar en caso de que sean ovstáculos. Esta información será recogida en el documento `grafo.txt` final, mostrado en el Anexo A.5.

3.3.2. Funciones destacadas en la implementación

Se considera primordial dar una breve descripción de las funciones más importantes que han permitido alcanzar el objetivo de particionar las imágenes y consecuentemente construir grafos contenidos en documentos de texto (`.txt`) con la información extraída de los mapas.

`binariza(int umbral)`

En el Fragmento de código A.3 se muestra como partiendo de una matriz que contenga los tonos del gris que representan la imagen, habiendo establecido previamente un umbral, todos los valores contenidos en ella quedarán binarizados mediante esta función. Para ello se realiza un recorrido a través de ellos, además de una comparación con la cifra umbral. Tomando en consideración el resultado de la comparación cada píxel será considerado *obstáculo* (negro = 0) o *vía libre* (blanco = 255).

La función `Celda::checkocupada()` también se basa en recorrer completamente, aunque en este caso sea la celda. Si se detecta en ella un obstáculo pasará a ser `true` la variable `bool Celda::ocupada`.

`escribirArchivo(vector<Celda>listaCeldas)`

La función mostrada en el Fragmento de código A.4 ha de servir para construir un archivo de texto, y de completarlo con los parámetros que son necesarios para que el programa ocupado de su lectura pueda generar una ruta para el robot móvil.

Es fundamental el cerrar los archivos después de usarlos, especialmente si se va a realizar un uso grande de memoria reservada. Esto puede generar un uso indebido del almacenamiento, que en caso de adquirir una magnitud notable puede ocasionar un funcionamiento indebido de la máquina a emplear.

3.3.3. Ampliaciones de los algoritmos

Para poder ampliar las posibilidades que ofrecen las implementaciones se han desarrollado una serie de extensiones que, partiendo de los resultados de las descomposiciones, introducen determinadas modificaciones a ellos persiguiendo una determinada finalidad. Estas ampliaciones serán presentadas brevemente en este apartado, mientras que sus efectos sobre los ensayos serán explicados en el Capítulo 4.

Agrupación de obstáculos: `agrupacion.cpp`

La finalidad de este esta ampliación es la de agrupar celdas que representen obstáculos en patrones cuadrados (de $2 \times 2, 3 \times 3, \ldots, n \times n$) de manera que juntas puedan conformar una única celda. Se realizará una lectura de un grafo ya desarrollado y una agrupación de grupos de obstáculos que formen un cuadrado/rectángulo más grande entre ellos.

3.4. Resultados de la descomposición

Los algoritmos de descomposición tienen que ser transmitidos a un programa que calcule la ruta de un robot a través de las celdas libres que se encuentren en ellos. Se ha escogido para la transferencia de información entre programas un formato de documento

de texto, que a su vez puede ser contenido en un mensaje de la plataforma ROS (Anexo A.2.3). Este documento debe contener toda la información que le permita al robot conocer la celda a través de la cual se mueve.

Se ha escogido como contenedor de la partición un grafo no dirigido. La etiqueta que corresponde a cada celda es la que se ha empleado como vía de acceso a la información contenida en cada celda. Se trata de un grafo no dirigido, ya que los movimientos de un robot que esté situado en una celda pueden ser realizados en la dirección opuesta. Por ello no se ha decidido restringir las aristas (relaciones) entre nodos (cada una de las celdas). Un ejemplo de ello puede verse en la Figura 3.8.

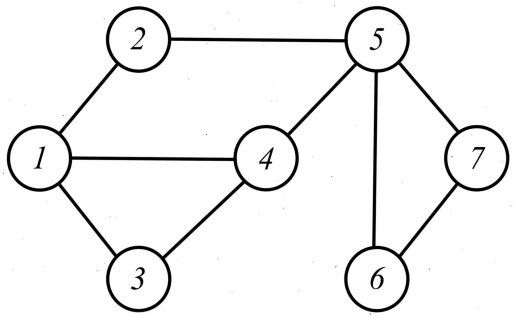


Figura 3.8: Grafo no dirigido.

El grafo generado en un archivo de texto contiene en su primera línea el formato en el que se van a dar sus datos, mostrado en el Fragmento de código 3.4. En la segunda línea aparece el número total de celdas, ocupando posteriormente cada una de ellas una línea. El significado de cada una de ellas, a pesar de quedar claro a través de su lectura, se desarrollará en el Anexo A.5.

```
1 identificador; ocupado; numero de vertices; num_x; lista_x;num_y;
    lista_y; numero de celdas adyacentes; identificador celda adyacente(
    puntos en comun):(primer punto)...(ultimo punto)
```

Fragmento de código 3.2: Leyenda de las líneas de cada grafo que representan a cada celda.

Capítulo 4

Estudio comparativo de los algoritmos implementados

Una vez definidos los métodos empleados para generar los grafos, se procede a comparar los algoritmos implementados. Para ello se ha planteado un modelo de regresión sobre el uso de los diferentes algoritmos generados en una colección de mapas.

En este capítulo también se van a analizar las cualidades de los mapas que permiten al robot llegar a su destino de una manera óptima siguiendo una metodología análoga.

4.1. Selección de las muestras

Conocer la influencia de diferentes variables sobre las características a estudiar a partir de métricas es fundamental seleccionar una serie de muestras que vayan a permitir extrapolar los resultados del estudio a otras topologías con las que compartan ciertas cualidades.

4.1.1. Límites de las muestras escogidas

El establecimiento de los límites del estudio a través de ciertas condiciones que se han considerado o escogido a partir del ámbito de desarrollo del trabajo es lo primero a realizar a la hora de diseñar una serie de experimentos.

Tamaño del mapa

Influenciado por el hecho de que el mapa representa un entorno 2D fuertemente abstracto en el que se han de mover los robots se ha decidido imponer una limitación de las dimensiones máximas de los mapas a 100 píxeles. Esta decisión queda además revalidada por las características didácticas de este proyecto, que hacen más difícilmente observables las cualidades de un mapa si sus dimensiones son excesivas.

Porcentaje de ocupación por obstáculos y el grado de aislamiento de los mismos

A partir de los mapas para la navegación de robot encontrados, los obstáculos no eran los elementos predominantes en las regiones internas a las delimitaciones de las estancias o entornos en los que habían de moverse. Por esta razón se ha impuesto un porcentaje de ocupación máximo de un 50 % del mapa.

Dado que se desea estudiar la influencia de la interconexión de los obstáculos en el posterior trazado de rutas, se define como obstáculo *aislado* aquel que tenga al menos otro píxel que represente un obstáculo con el que comparta caras, o aquel que esté compuesto por un único píxel. La tipología de obstáculo opuesta a ello será la de un obstáculo *conectado*. Con este último se hace referencia a todo tipo de relaciones entre obstáculos.

4.1.2. Caracterización de los grupos de estudio

Se han generado 12 diferentes clases que permiten categorizar las muestras según sus cualidades. Se representan mediante el siguiente código:

Mapa X Y Z

Donde cada uno de los componentes de esta codificación tendrá una característica asociada a un carácter o cifra.

Se ha desarrollado un generador de imágenes `.pgm` escrito en C++ que además permite modificar en los mapas su tamaño, ocupación y grado de aislamiento de los obstáculos. Para no afectar los resultados con sesgos indeseados y con objeto de asegurar la calidad de los resultados se ha utilizado un generador de números aleatorios para la producción de nuevos mapas. Ejemplos de ellos se pueden encontrar en el AnexoA.4.

X - Tamaño

- A - Pequeño: $Longitud_{Lados} \leq 33$ píxeles
- B - Mediano: 33 píxeles $< Longitud_{Lados} \leq 67$ píxeles
- C - Grande: 67 píxeles $< Longitud_{Lados} \leq 100$ píxeles

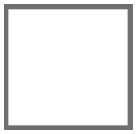


Figura 4.1: Mapa A Y Z

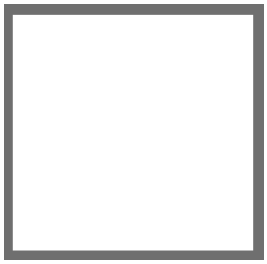


Figura 4.2: Mapa B Y Z

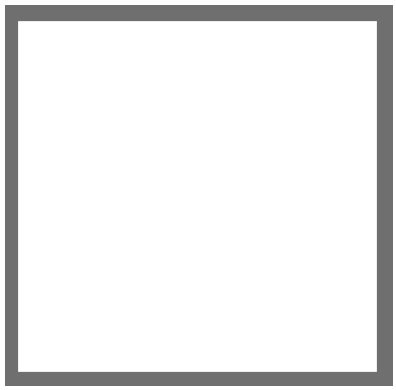
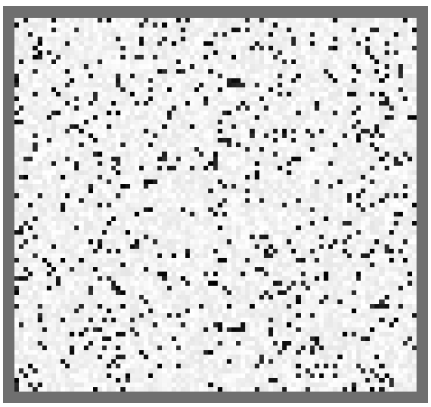


Figura 4.3: Mapa C Y Z

Y - Ocupación

- 0 - Poco ocupado: $Porcentaje_{obstaculos} \leq 25\,\%$
- 1 - Muy ocupado: $25\,\% < Porcentaje_{obstaculos} \leq 50\,\%$



(a) Mapa X 0 Z

(b) Mapa X 1 Z

Figura 4.4: Mapas diferenciados según su ocupación.

Z - Grado de aislamiento de los obstáculos

- 0 - Obstáculos aislados
- 1 - Obstáculos conectados

(a) Obstáculos aislados (b) Obstáculos conectados

Figura 4.5: Diferentes obstáculos.

4.2. Metodología de la comparación

A la hora de estudiar la adecuación de determinado tipo de mapa o método de descomposición del mismo en celdas se han empleado determinadas métricas, también denominadas indicadores clave del rendimiento (KPI)[1]. A partir de ellas, y de las características estadísticas de sus mediciones se construyen una serie de figuras de mérito[2] con objeto de facilitar su análisis.

A fin de conseguir una mayor representatividad de los resultados obtenidos se han generado 10 mapas para cada uno de los 12 grupos de estudio. Un ejemplo de cada uno de ellos se muestra en el Anexo A.4. A cada uno de estos mapas se le aplican los dos algoritmos de descomposición implementados con una dimensión mínima de partición de 2 píxeles, obteniendo de ellos las métricas desarrolladas en el Apartado 4.2.1.

Se extraen 5 KPIs de cada una de las simulaciones mostradas en el Anexo A.7. De cada uno de los resultados se presentan además el valor medio y la desviación típica en los resultados expuestos en el AnexoA.7.

[1]Las siglas KPI proceden de las iniciales de su traducción al inglés *Key Performance Indicator*.

[2]Una figura de mérito es una cantidad utilizada para caracterizar el rendimiento de un dispositivo, sistema o método, en relación con sus alternativas [15].

4.2.1. Magnitudes estudiadas y sus correspondientes métricas

Cuando se trata de dictaminar el comportamiento de un robot prima que su actuación sea rápida y precisa. Por ello se han evaluado las siguientes KPIs mediante simulaciones de la aplicación de los algoritmos desarrollados sobre muestras aleatorias.

Tiempos (t)

La medición de los tiempos de realización de cada uno de los procesos es una herramienta fundamental en el estudio de las propiedades de un software. Será ventajoso cuanto menor sea el resultado, que van a ser dados en milisegundos, dado que el método de medición (Anexo A.6) así lo permite. Se trata de obtener así una mayor precisión. Las variables a estudiar son:

- t_{desc} **:** Tiempo de generación de las descomposiciones a partir de un mapa.
- t_{ruta} **:** Tiempo de generación de una ruta entre dos celdas aleatorias[3] del grafo generado por la descomposición.

Longitudes (L)

Existen determinadas distancias que se consideran fundamentales para evaluar el desempeño de un algoritmo de descomposición, o del empleo de un determinado mapa para la posterior generación de rutas a través de él. La unidad escogida es la medida de los píxeles. Esta es una cantidad abstracta, ya que estos pueden ser asignados a cualquier dimensión real en función del campo de aplicación del mapa o de ls precisión deseada en la abstracción de obstáculos reales. Las variables a estudiar son:

- S_{celda} **:** Superficie media de las celdas del grafo[4].
- $A_{ruta} = \frac{L_{diagonal}}{L_{ruta}}$ **:** Adecuación de una ruta generada: Permite comparar cuan corta es la ruta a través de los centroides de las celdas escogidas en el cálculo del tiempo de ruta (t_{ruta}) frente a la recta que una sus centroides.

[3]Estas dos celdas han de ser libres y que entre ellas exista una posible ruta.

[4]A pesar de que esta métrica no aporte ningún valor para la descomposición en retícula (queda impuesta por la dimensión escogida para el lado de la cuadrícula en las simulaciones), permite evaluar la variabilidad de los tamaños de las celdas en la descomposición rectangular. De esta forma sería fácilmente comparable con cualquier otra metodología de descomposición en celdas del mapa.

Adecuación de los porcentajes de ocupación (A_{ocup})

Dado que se experimenta con descomposiciones en celdas no exactas, nos encontramos con que las celdas generadas que representen un obstáculo no siempre están conformadas totalmente por obstáculos. Se compara de por lo tanto el porcentaje de ocupación del mapa entre el porcentaje de ocupación del grafo por celdas consideradas como obstáculo. El resultado se da como un número decimal con tres cifras significativas, que siempre es inferior o igual a 1, solo en caso de que la descomposición capte los obstáculos como celdas de manera exacta.

La variable a estudiar es:

- $A_{ocup} = \frac{Oc_{mapa}}{Oc_{grafo}}$ **:** Adecuación de una determinada descomposición a la hora de denominar a las celdas como obstáculos.

4.3. Presentación y análisis de los resultados

En este apartado se exponen las características los resultados de las comparaciones realizadas. A partir de ellas se analiza el funcionamiento de las descomposiciones implementadas en las diferentes tipologías de mapas. Todo ello se realiza empleando las métricas calculadas en las simulaciones, cuyos resultados se exponen en el AnexoA.7.

En los cálculos de los tiempos, los resultados obtenidos no los considero como totalmente representativos. Esto se debe a que la máquina empleada en su cálculo se encontraba en situaciones muy diferentes entre una simulación y otra, con numerosos procesos funcionando en paralelo en las diferentes situaciones de cálculo. Para evitarlo hubiera sido mejor el contar con una máquina exclusiva para la realización de los ensayos, cosa que ha sido imposible por los medios materiales disponibles en la realización del proyecto. De todas maneras se han podido apreciar determinadas dinámicas de los resultados que serán mencionadas a continuación.

Tiempos de descomposición

En la Figura4.6 se aprecia como los tiempos tardados para producir el grafo de celdas en el programa que contiene al algoritmo de descomposición en retícula se ajustan linealmente al tamaño de los mapas empleados. Esto se debe a que divide el mapa en cuadrados de dimensión constante, analizando si alguna de las celdas que lo componen

son obstáculo, pasando a ser considerada la celda entera como un obstáculo.

Debido al procedimiento de obtención de un grafo a partir del algoritmo de la descomposición rectangular esta dependencia del tiempo no es en absoluto lineal. Entran en juego otras dependencias mas allá de las dimensiones del mapa.

Una de los comportamientos observados de los tiempos de generación de la descomposición rectangular está muy relacionado con la superficie media de la celda de cada descomposición: cuanto mayor sea S_{celda}, esto implicará que será menor el número de particiones recursivas en *quadtree*. Esto a su vez significará que el tiempo en ejecutar el programa será menor.

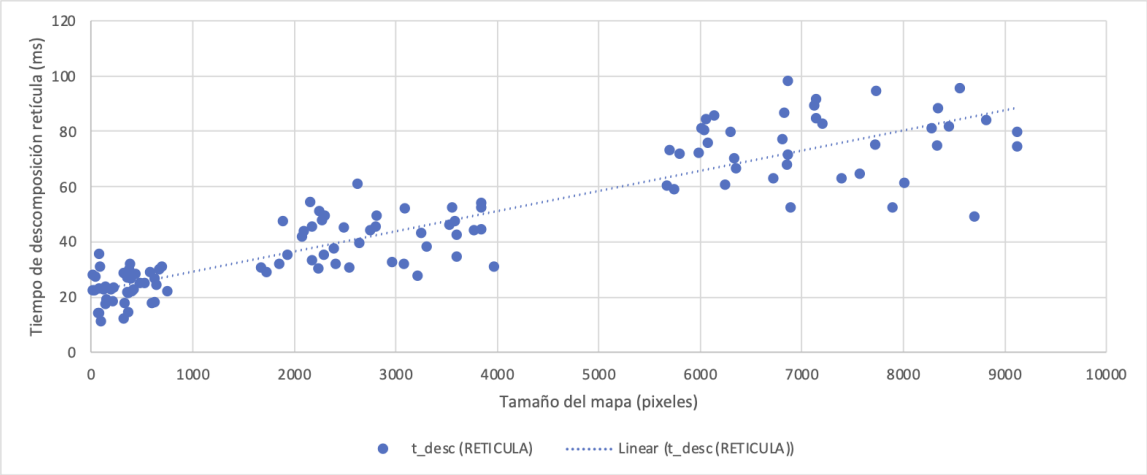


Figura 4.6: Tiempos de generación de una descomposición en retícula frente a los tamaños de las celdas.

Superficie media de las celdas

Debido a los fundamentos de la descomposición, el método de descomposición reticular da unas celdas de unas dimensiones constantes: el cuadrado de la definición escogida (Número de píxeles que forman el lado de cada uno de los cuadrados). Para obtener resultados equiparables en las demás métricas se ha tomado el mismo valor de dimensión para las celdas dela descomposición en retícula como la tolerancia o dimensión mínima del borde de celda en la descomposición rectangular: 2 píxeles.

Tal y como se ha dicho, los tiempos de generación de las descomposiciones rectangulares siguen una tendencia de ser inversamente proporcionales al tamaño medio de las celdas de los grafos generados por estos (Figura4.7.

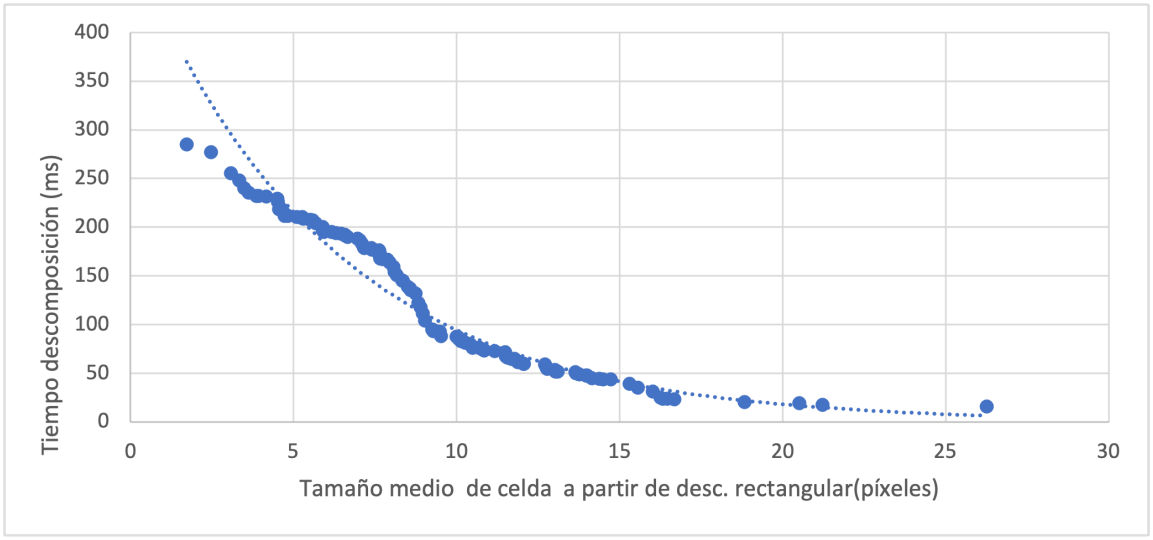


Figura 4.7: Tiempos descomposición rectangular en función de la Superficie media de las celdas.

Para presentar las propiedades de los resultados se emplea la tipología de gráfica mostrada en la Figura 4.8. Esta gráfica es denominada como gráfica de cajas y bigotes, y muestra tanto la media como la mediana, dando información de los cuartiles así como de la aparición de datos que no sigan la distribución típica. Estos últimos probablemente provengan de ensayos que no hayan sido realizados adecuadamente.

En estas gráficas de cajas se puede ver como se obtienen mejores resultados para los subgrupos de mapas poco ocupados por obstáculos y cuyos obstáculos se agrupen entre si.

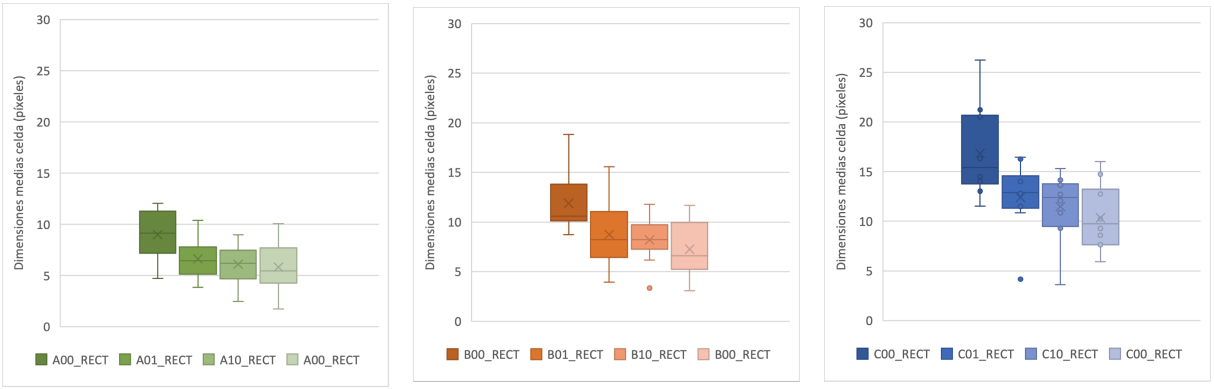


Figura 4.8: Dimensiones medias de las celdas de las descomposiciones rectangulares.

Adecuación de los porcentajes de ocupación

Según lo observado a partir de los análisis, y teniendo en cuenta el método a través de l cual han sido programados los algoritmos de descomposición, la adecuación de la adaptación será mayor para casos de estudio en los que las celdas estén formados por obstáculos aislados (Mapas X Y 0). Esto se debe a la habitual mayor ocupación de las celdas consideradas como obstáculo de estos métodos exactos de descomposición para dicho grupos de estudio (al encontrarse con mucha más probabilidad un pixel que represente un obstáculo rodeado por otros obstáculos).

Para realizar la comparativa entre los dos métodos de descomposición se emplea de nuevo el diagrama de cajas de la Figura 4.9. En ella se observa como para mapas más grandes existe menor variabilidad. El hecho de que los resultados obtenidos para las descomposiciones rectangulares sean similares a los obtenidos para descomposiciones en retícula hace pensar que esta no es una buena figura de mérito para la comparativa entre los métodos implementados.

Lo que si que puede extraerse de las gráficas de los mapas de tamaño grande en la descomposición rectangular, y corresponde a lo esperado es como los mapas con obstáculos agrupados son en cierto modo mejores que los de obstáculos dispersos a la hora de ser descompuestos en celdas.

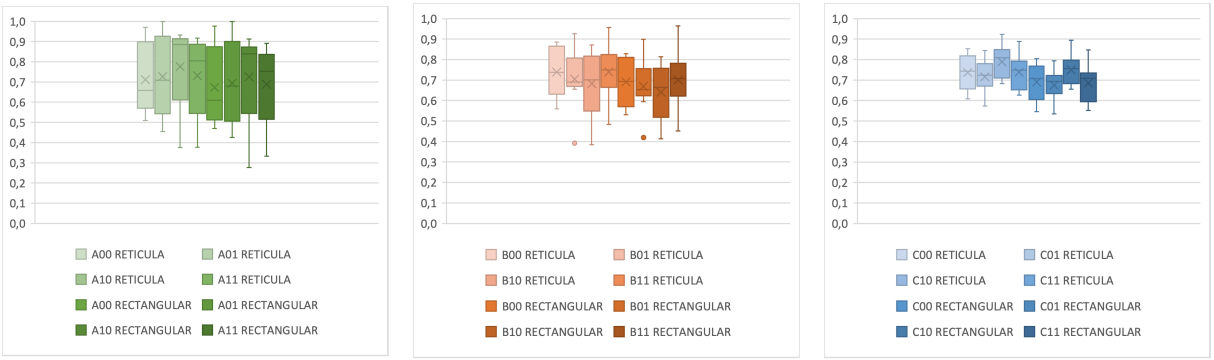


Figura 4.9: Adecuación de los porcentajes de ocupación

Adecuación de la ruta generada

Esta figura de mérito que compara la distancia entre los centroides de las celdas seleccionadas aleatoriamente y las distancia del recorrido realizado por el robot permite hacernos a la idea lo bien que se ajusta a su finalidad cada uno de los algoritmos de descomposición implementados.

Gracias a su mayor granularidad, los resultados obtenidos en la descomposición en retícula dan una buena base para que el robot pueda trazar la ruta de la forma que mas se ajuste a la diagonal (la vía más corta). A cambio, sus tiempos de generación de ruta (A_{ruta})serán superiores a los de la descomposición rectangular.

De todas formas, el método de descomposición rectangular genera una rutas bastante similares. Presentan una mayor variabilidad debido a los tamaños variables de celdas principalmente. Estos resultados pueden verse en la Figura A.2.

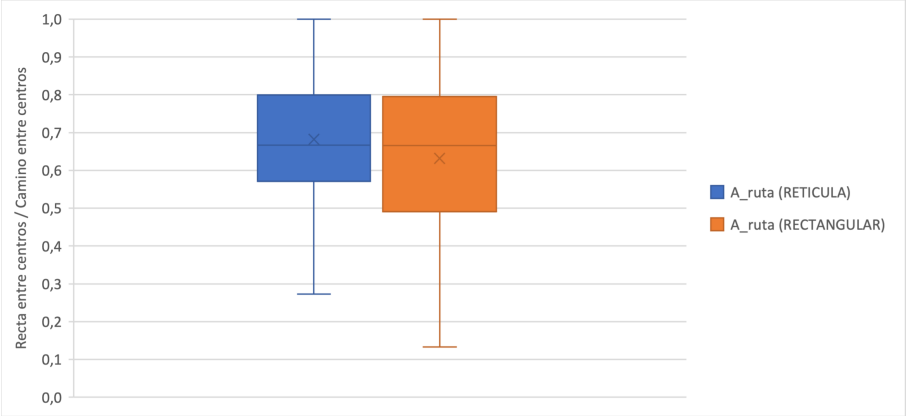


Figura 4.10: Adecuación del trazado de rutas generadas a partir de las celdas contenidas en grafos de de composición.

Se consideran significativos de mención los resultados obtenidos de la Adecuación de las rutas (A_{ruta}) iguales a la unidad. Estos han ocurrido en los casos en los que las celdas a unir se encontraban en una línea recta, o bien eran adyacentes.

Tiempos de trazado de ruta

Estos tiempos de generación de rutas son muy superiores a los de generación de descomposiciones en celdas tomando como referencia los tiempos tratados en este trabajo.

La principal razón es que el algoritmo empleado en su trazado está basado en un análisis de todas las celdas contenidas en el grafo (Algoritmo de Djikstra, AnexoA.8.2).

Según lo expuesto, los tiempos tardados en generar una ruta (t_{ruta}) tienen una correlación proporcional con el número de celdas libres contenidas en el grafo. Por lo tanto, en las descomposiciones rectangulares existe una cierta dependencia de las dimensiones del mapa (A_{ocup}). Ello queda representado en la Figura4.11 comparándolo a los tiempos de generación de ruta a partir de descomposiciones en retícula, en los que la relación crece de una forma más rápida con el tamaño del mapa.

Se puede ver también en dicha gráfica como los resultados son menos dispersos en la descomposición rectangular, al existir un numero de celdas más acotado.

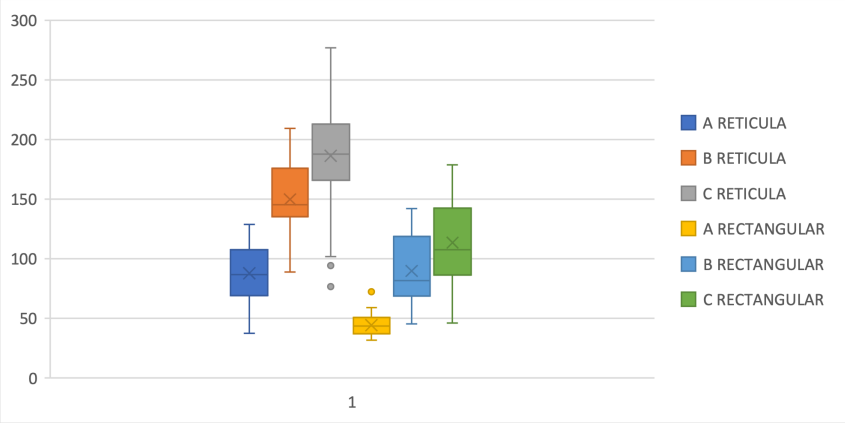


Figura 4.11: Tiempos de generación de ruta en función del tamaño del mapa y del método usado en su descomposición.

Cabe mencionar que no se ha de relacionar directamente el tiempo de generación de rutas a partir de un grafo generado por la descomposición rectangular con el tamaño de los mapas, ya que la metodología empleada para generar rutas se basa en un rastreo iterativo en todas las celdas libres a través de las cual podría circular el robot, y el número de celdas se ha comprobado que para las descomposiciones rectangulares se ve afectado por muchas más variables que por las dimensiones del mapa.

Por último es interesante mencionar que gracias al refinado de los algoritmos implementados estos pueden mejorar sus resultados temporales así como sus capacidades. Desarrollos más allá de los ya implementadas han quedado fuera de las dimensiones de este trabajo debido al gran [5]esfuerzo temporal y material que requerirían para resultar realmente ventajosas.

[5]Basándose en lo requerido para desarrollar lo que ya ha sido implementado.

Capítulo 5

Conclusiones

Como epílogo se puede extraer del trabajo realizado la importancia que tiene para la labor de orientación de los robots móviles el desarrollo de algoritmos que les faciliten la información contenida en los mapas.

Cabe destacar que el estudio comparativo sirve no solo para analizar las características del comportamiento de los algoritmos sobre los mapas generados, sino también como base de un proceso de innovación a partir de la generación de nuevos algoritmos, cuyo funcionamiento puede ser caracterizado gracias a las métricas estudiadas, y posteriormente comparado a otros gracias a las figuras de mérito presentadas.

Entre los algoritmos implementados prevalece sin duda la descomposición en celdas rectangular, esencialmente gracias al potencial que le confiere su propiedad de recurrencia. En el único parámetro en el que es claramente inferior a la descomposición en retícula es en tiempos de generación de las descomposiciones. Este parámetro es probablemente el menos crítico de los tratados, al poder ser los grafos generados de antemano partiendo de mapas conocidos.

A partir de los mapas empleados y del estudio de las métricas seleccionadas se puede inducir que las particiones en celdas no exactas son más eficaces en su función de extraer la información del mapa cuando estos mapas presentan obstáculos aislados y bien definidos.

En caso de que los mapas estén repletos de obstáculos las descomposiciones más adecuadas serán las que sean más "inteligentes" a la hora de reconocer los grupos de obstáculos y las relaciones que guardan entre ellos.

Para llegar a poder exponer estas conclusiones ha sido necesaria una robusta formación en el campo de la robótica. Gracias a los fundamentos de Informática aprendidos en el Grado en Ingeniería Industrial, así como de las teorías de control en las que está basada la robótica móvil se han añadido los conocimientos específicos precisos para el desarrollo de este trabajo. Además, el desarrollo de numerosos tutoriales ha permitido aplicar los conocimientos aprendidos. El desarrollo de gran parte del software ha resultado una parte fundamental para el aprendizaje y la interiorización de los conocimientos.

Cabe mencionar alguno de obstáculos encontrados, que han sido superados para el desarrollo de este trabajo. Una incompatibilidad entre la versión del sistema operativo del ordenador empleado para el desarrollo de los algoritmos (MacOS) y el programa VirtualBox, donde se encontraba la máquina virtual con Ubuntu ha sido un inconveniente encontrado a comienzos del trabajo.

Se ha decidido escribir los algoritmos en el editor de código Virtual Studio Code para MacOS por recomendación de expertos en el ámbito de la programación de software, así como por su uso intuitivo. Posteriormente se ha verificado el adecuado funcionamiento de los algoritmos implementados en ROS gracias a un contenedor virtual del sistema operativo Ubuntu incorporado en la propia aplicación VS Code denominado Docker. Una explicación más detallada de todas estas herramientas informáticas puede encontrarse en el Anexo A.2.

En varias ocasiones han aparecido problemas en la implementación del código. Para resolverlos se han depurado los errores siguiendo metodologías de *debugging*[1]. Se ha logrado así realizar las correcciones pertinentes para el adecuado funcionamiento de los programas.

De cara a la aplicación del trabajo en proyectos de robótica se le pueden y deben aplicar diferentes ampliaciones. Por ejemplo, además de la información contenida en los grafos desarrollados hay otros datos que pueden ser añadidos a ellos, como la selección de determinadas zonas de interés o de otras distinciones más allá de obstáculos o vías libres. También se pueden ampliar las capacidades de reconocimiento de estructuras o formas dentro de los mapas escogidos. Estos objetivos no han sido considerados debido a la magnitud de la ampliación del trabajo que requieren.

[1]Término internacional proveniente de la lengua inglesa que hace referencia al proceso de corrección de errores en la escritura del código fuente.

Apéndice A

Anexos

A.1. Estado actual de las tecnologías relacionadas

En este anexo se analiza el contexto en el que están desarrollados los diferentes ámbitos tratados en la memoria, al igual que los conceptos empleados o mencionados en los capítulos de la memoria. El enfoque de esta sección parte de los conceptos más generales y se especializa paulatinamente en lo más relevante del ámbito proyecto mediante la descripción de los instrumentos más destacados.

A.1.1. Robótica

Al tratarse la robótica de un campo inmensamente amplio y multidisciplinario afectando a numerosas ramas del conocimiento, los avances producidos que guardan relación con robots son casi continuos. Esto hace que la robótica haya cobrado una inmensa importancia en el mundo actual. Este área de la ciencia suele ser englobada por los conocimientos cubiertos en la ingenierías informática, mecánica o electrónica entre otras. Han de ser consideradas también otras áreas de conocimiento que traten de una fina y segura colaboración entre personas y robots [16], así como de la ética de las relaciones entre el ser humano y la inteligencia artificial.

Dentro de la robótica se explican determinadas tipologías que pueden emplear la descomposición en celdas como metodología parcial para lograr sus objetivos:

Robótica de servicios

La robótica de servicios puede ser definida como la que cubre lo relacionado con los robots que realicen las tareas originalmente hechas por un humano en un área específica. Estos robots no eran autónomos, sino que han servido como herramientas que faciliten o reduzcan riesgos en el trabajo. En cambio hoy en día estos robots son capaces de realizar cada vez más y más tareas de forma independiente como se narra en la Sección A.1.2

La autonomía de los robots empleados en servicios es actualmente uno de los objetivos más perseguidos, permitiendo a los robots ser empleados en diversos ámbitos como las cadenas de producción, envasado, o la limpieza y otros numerosos campos. El principio de diseño ha de ser modificado minuciosamente para cada aplicación. Por ello es importante desarrollar una sólida organización del uso de los robots en servicios a través de una estructura modular y una accesibilidad a la orden del día [17].

(a) Limpieza

(b) Líneas de producción

Figura A.1: Diferentes aplicaciones de robots en servicios.

Imágenes de uso libre.[18]

Robótica móvil

Los robots con la capacidad del desplazamiento han de tener la habilidad de orientarse y moverse de manera autónoma, con la suficiente inteligencia como para reaccionar de forma que puedan evitar los obstáculos del entorno en el que se han de desplazar. Para ello es necesario que dichos robots obedezcan unos algoritmos establecidos cuando fueron programados. En este aspecto del movimiento en un entorno resulta fundamental comprender los métodos que se emplean para generar las diferentes rutas a través de las partes del mapa. Estas partes representan los espacios por los cuales los robots podrán moverse, o bien sean un obstáculo o área de interés.

Pero antes de generar una ruta, un robot ha de conocer el entorno en el que ha de trazarla empleando sensores. Estos instrumentos mayoritariamente están basados en la captación y tratamiento de los mismos impulsos que los seres humanos: acústicos, visuales o térmicos [19]. De todas formas, muchos robots emplean como base donde planificar sus trayectorias planos del entorno en el que se van a desplazar. Posteriormente estas representaciones deben de ser simplificadas en estructuras de datos mediante determinados algoritmos (como los que se desarrollan en este trabajo) para así generar las rutas.

En los métodos de descomposición de mapas en estructuras de datos no es único el de la descomposición en celdas. Existen muchos otros procedimientos, habitualmente utilizados en una aplicación más específica de la robótica móvil. De esta forma se alcanza mayor profundidad y precisión en las soluciones. Estos otros tratamientos de los mapas están basados en diversas tecnologías, como herramientas de la visión por computador con reconocimientos avanzados de formas u otras técnicas basadas en la inteligencia artificial [20]. Debido a su especificidad no son fácilmente replicables en otras aplicaciones. En cambio el método de descomposición en celdas narrado en esta memoria resulta ventajoso en este aspecto, dada su enorme versatilidad.

Figura A.2: Robot móvil autónomo de tipo AGV empleado en plantas industriales.

A.1.2. Robótica autónoma

La robótica autónoma es la rama de la robótica que trata lo relativo al funcionamiento de robots de forma independiente. En consecuencia los robots autónomos realizan por si mismos las tareas de orientarse, desplazarse y de actuar, todo ello sin la intervención humana. Para que esto pueda ser llevado a cabo se ha de lograr un equilibrio y colaboración entre las numerosas tecnologías que forman parte del robot.

Los robots autónomos toman los datos en los que basan sus acciones de dos principales fuentes: de datos del entorno, como pueden ser los mapas o la información proporcionada por sensores, o de datos internos, calculados a partir del propio funcionamiento del robot [21]. Un ejemplo de esto es la navegación por estima, que emplea el posicionamiento a través de la odometría, explicado en la Sección A.1.3.

A.1.3. Tecnologías de la orientación en la robótica móvil

A lo largo de esta sección se introducen diferentes técnicas propias de la robótica móvil empleadas para localizar al robot, desarrollar un mapa del entorno y trazar rutas en él para que así se pueda alcanzar el objetivo.

Posicionamiento y odometría

Es necesario conocer determinados conceptos para la localización de un robot: su posicionamiento inicial, así como la odometría, que es la información que permitirá estimar su propia posición a lo largo de su navegación al robot.

Para localizar un robot se precisa de la generación de un mapa y en caso de que el posicionamiento del robot y el mapa sean desconocidos se puede generar un mapa a partir de la información generada por los sensores del robot. Esta técnica, fuera del ámbito de este trabajo, se denomina SLAM: siglas de la expresión inglesa "*Simultaneous Localization And Mapping*" (localización y mapeo simultáneos).

Para rastrear los movimientos de los robots de manera interna se usan *encoders*, es decir, dispositivos electro-mecánicos acoplados a los ejes de las ruedas de los robots, cuyas revoluciones pueden ser traducidas en un desplazamiento lineal relativo al suelo. También se pueden emplear otros métodos de seguimiento basados en telecomunicaciones, como puede ser el sistema de posicionamiento global (GPS). Estas fuentes externas de información tienen habitualmente de una precisión mucho menor, además de generar un coste añadido.

Al tratar de situar al robot en su entorno se generan errores que impiden la correcta localización del mismo. Estos errores pueden dividirse en dos categorías: los errores sistemáticos, que se cometen en todas las mediciones que se realizan de una determinada magnitud, y los errores no sistemáticos, que derivan de un mal funcionamiento de los sistemas de localización propia debido a un ajuste inadecuado de estos con el entorno que los rodea.

Los errores sistemáticos son los errores que derivan de problemas en la calibración de los sensores o de partes del robot [22]. Los errores no sistemáticos provienen de una mala adaptación al entorno, como por ejemplo el deslizamiento de las ruedas del robot con una superficie en la que hay aceite. En esta situación el encoder registraría unas rotaciones de las ruedas que debido al deslizamiento no se corresponderían con el movimiento real del robot.

Se ha de tratar reducir los errores al mínimo para un correcto posicionamiento del robot. Se ha de hacer especial énfasis en los errores sistemáticos, ya que al acumularse de manera constante generan por tanto una distorsión importante entre la realidad y la información que percibe el robot.

A.1.4. Mapeo

Una vez el robot ha sido localizado y conoce su situación para con el entorno que lo rodea, un mapa ha de ser generado y el robot ha de orientarse en él de tal forma que pueda alcanzar su objetivo de la manera con menos impedimentos. De esta forma se conseguirá una mejora sustancial frente a las posibilidades que podría dar un mapeo no planificado, en el cual el robot construiría su camino a partir de los impulsos externos que fuese recibiendo.

Existe un enfoque del mapeo que parte de un mapa desconocido, para el que ha de generar un modelo de caja negra en el que, a partir de la información de los sensores, se ha de generar una estructura y los parámetros que permitan maniobrar en ella. Este modelo es el denominado como SLAM, ya mencionado en el anterior sub-apartado Posicionamiento y odometría.

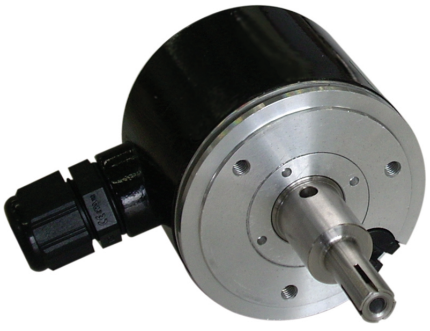


Figura A.3: Encoder rotacional.

Imagen de uso libre.[18]

Se ha prescindido de una mayor indagación en esta temática debido a la dimensión y alcance de este trabajo, a pesar de las incontables posibilidades ofrecidas a través de la plataforma ROS, así como del inmenso desarrollo actual.

Existe una infinidad de métodos de dividir imágenes que representen mapas. Por ello han de primar los que hagan la posterior labor de generación de rutas de la manera más sencilla para el robot. A continuación se describen los principales conceptos en relación a los mapas que serán tratados en la memoria.

Mapeo topológico

Un mapa topológico es aquel en el cuál se ha prescindido de toda la información que pueda ser considerada como trivial para su posterior uso. Por eso se va a tratar de tomar como objetivo esta tipología, ya que resulta importante que los mapas a desarrollar no contengan datos superfluos que generen un acceso erróneo a la información.

Mapeo métrico

A la hora de generar un mapa es importante establecer una equivalencia entre las medidas del mapa y las medidas de la realidad representada. Para ello es fundamental evitar errores sistemáticos en la toma de medidas por los sensores. las medidas se traducirán en las distancias que finalmente habrá de recorrer el robot, que han de ser lo más pequeñas posible. Los mapas métricos carecen de significado y relevancia más allá de las medidas que contienen. Se muestra un ejemplo en la Figura A.4.

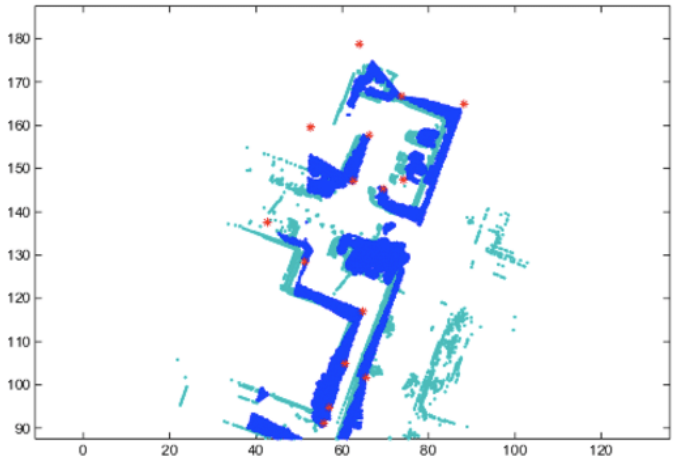


Figura A.4: Mapa métrico de un entorno de interiores generado por un robot.

Extraído del documento *"Aportaciones a la auto-localización..."* de F.Martín [23]

Mapeo híbrido

El mapeo híbrido consiste en aunar las ventajas que tienen los diferentes procedimientos que puede seguir un robot para interpretar el medio en el que se mueve. Por ende se busca un mapa topológico ajustado mediante una escala con medidas reales siendo además enriquecido semánticamente, es decir, por las relaciones que guardan entre sí los diferentes elementos del mapa.

El mapeo basado en redes de ocupación (*occuppancy grids*) representa el tipo de mapeo a desarrollar en este trabajo. En él se contiene un diagrama del plano con una serie de obstáculos y de vías para que circule el robot. Esta tipología de mapa se ha escogido no solo por que lleve siendo uno de los predominantes en el campo de la robótica desde los inicios de esta, como es nombrado en el artículo de A.Elfes escrito en 1987 [24], sino también a que es lo suficientemente general como para contener a los algoritmos desarrollados en este trabajo. De esta manera se obtiene un método didáctico con gran libertad de aplicación.

A.1.5. Trazado de rutas

La generación de caminos para alcanzar un destino es la principal utilidad de los mapas generados. Por esta razón se ha de conocer los fundamentos de la composición de vías para la circulación de los robots.

Existe una gran variedad de estrategias para generar rutas para un robot a través de un mapa. Se pueden distinguir dos principales metodologías: las basadas en una lectura global del mapa y las que ocupadas de un sampleo local de los espacios, empleados preferiblemente en mapas de grandes dimensiones.

Tal y como expresa B.K. Patle en su artículo de 2019 acerca de las estrategias de planificación de rutas en el ámbito de la robótica móvil [25] se ha dado una transición en la actualidad hacia métodos de generación de rutas mixtos: a partir de las bases establecidas por los métodos clásicos ya mencionados, se han empleado numerosos métodos reactivos basados en redes neuronales para dar consistencia y versatilidad a los enfoques tradicionales.

A lo largo de este trabajo los métodos empleados para el trazado de rutas estan basados en una lectura de todas las celdas que componen un plano. Al no ser incluido el trazado de rutas en los objetivos del proyecto, si no que únicamente va a ser empleado para muestrear la utilidad de las descomposiciones de los mapas generados. Por esta razón el análisis del grafo completo se ha preferido frente un muestreo parcial, de manera que así se contemplen todas las celdas generadas.

A.2. Herramientas informáticas empleadas

Los robots captan y procesan datos, actuando consecuente de la forma que han sido programados. Por ello es fundamental describir el entorno informático en el que se desarrollan los programas que dictan las resoluciones que son tomadas para lograr un resultado deseado. Se van a exponer en este anexo todas las herramientas informáticas usadas a lo largo del proyecto, de una manera más detenida y haciendo hincapié en las facetas que más se han empleado de ellas.

A.2.1. Sistema operativo

Dada la gran variabilidad hay ningún sistema operativo estándar en la robótica: Cuando los robots tienen un acceso a un ordenador completo, suelen ejecutar algún sistema basado en Linux, que permite aunar todas las funcionalidades que requiere un robot, mientras que otros robots cuya finalidad es más limitada simplemente ejecutan utilizan plataformas de microcontroladores como la ya mencionada Arduino (Figura A.5b). También hay robots que precisan tiempos de respuesta bajos que emplean sistemas operativos en tiempo real.

Es bastante común que los robots grandes y complejos tengan varios ordenadores que ejecutan diferentes sistemas en función de sus necesidades: un ordenador principal que ejecuta un sistema GNU/Linux y se encarga de la orquestación de los procesos y de las tareas más lentas; las plataformas más ligeras y cercanas al hardware realizan tareas de control y de procesamiento motor de bajo nivel. [26]

Ubuntu (Linux)

Linux es un conjunto de sistemas operativos basados en el sistema operativo UNIX, como Windows o MacOS. En cambio, la familia de sistemas operativos Linux está basado en el desarrollo modular de código abierto. En estas últimas décadas este movimiento ha mostrado un gran crecimiento y aceptación por parte de la comunidad de desarrolladores informáticos.

La denominación de este conjunto de sistemas operativos como Linux hace referencia al nombre del creador de su *kernel* original, Linus Torvalds. Este contiene el núcleo del computador y se ocupa de las interacciones entre el software, así como entre el software y hardware. Una de las partes más destacadas es el motor gráfico, que permite al usuario la visualización de entornos.

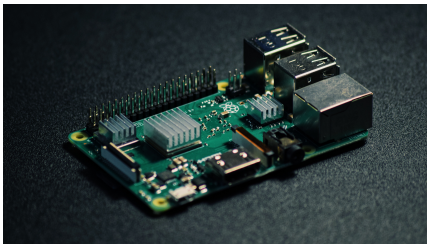
De todas formas son necesarios muchos otros elementos como memoria o periféricos como pantalla para asegurar el correcto funcionamiento de un ordenador.

Linux cuenta con una gran variedad de compiladores para la mayoría de los lenguajes de programación. A su vez, siempre pueden añadirse más desde el acceso a los mismos en repositorios. Gracias a esto, es comúnmente usado para el desarrollo de programas y aplicaciones, ya que brinda la posibilidad de desarrollarlos para cualquier plataforma. Para ello, utiliza un compilador cruzado, muy apreciado a la hora de realizar aplicaciones multiplataforma.

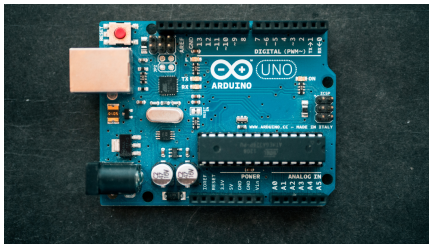
Los mencionados módulos y funcionalidades han sido agrupados en paquetes que pueden ser empleados, denominándose distribuciones o *distros*. Una de las más comunes y la que ha sido empleada en este proyecto es Ubuntu. La principal razón para su uso es que ROS está desarrollado exclusivamente para Ubuntu cuando comenzó este proyecto en septiembre de 2020.

A.2.2. Lenguajes de programación

Tanto Python como C/C++ son los idiomas más utilizados hoy en día en el mundo de la robótica. Se emplean lenguajes muy similares para escribir programas para dos de las herramientas de control más empleados hoy en día en la iniciación a la robótica: la placa base Raspberry Pi (Fig. A.5a) y el microcontrolador Arduino (Fig. A.5b) respectivamente. Se puede también usar código en ambos lenguajes para implementar programas en ROS, la mayor plataforma de software libre para proyectos de robótica. Se ha empleado C++ ya que es considerado como una de las formas más didácticas de ponerse en contacto con la robótica dada su modularidad y la dualidad de metodologías procedural y orientada a objetos.



(a) Raspberry Pi



(b) Arduino

Figura A.5: Principales modelos de hardware en la iniciación a la robótica.

Imágenes de uso libre en Unsplash [18].

C++

C ++ es un lenguaje de programación de forma libre de propósito general que admite la programación procedimental, orientada a objetos y genérica. Fue desarrollado por Bjarne Stroustrup en 1979 como una mejora del lenguaje de programación C, al que se le añadieron clases pudiendo así desarrollar también programas orientados a objetos.

C++ es empleado hoy en día por numerosos programadores en todo ámbito de aplicación: actualmente se está utilizando mucho para escribir controladores de dispositivos y otro software que se basa en la manipulación directa de hardware bajo restricciones de tiempo real al ser uno de los lenguajes con una menor demora en sus tiempos de respuesta. Cualquier usuario de un PC con Windows o de un ordenador Mac ha usado de forma indirecta C++, al estar desarrolladas en este lenguaje las principales interfaces de usuario de estos sistemas operativos.

A.2.3. Entornos de trabajo

En el ámbito de la informática se le denomina como entorno de trabajo al archivo, directorio o software que permite juntar determinados recursos digitales y trabajar con ellos como una unidad cohesiva [27]. A pesar de que también se denomine así, no se ha de confundir con el medio en el cual un robot ha de realizar su trabajo, o en el caso que incumbe a este proyecto, en el cual ha de desplazarse.

El entorno más empleado en el ámbito de la robótica móvil es ROS. La idea básica que hace de ROS un sistema tan útil a la hora de desarrollar programas orientados a la robótica es la posibilidad de desarrollar numerosos procesos diferentes en varias secciones denominadas nodos.

Además, también se emplea en numerosos proyectos de toma y análisis de datos el software Matlab de análisis matemático. Cabe destacar las posibilidades que ofrece en la ingeniería de control gracias a la representación de funciones de transferencia mediante la herramienta propia de Matlab denominada Simulink y la generación automática de código para microcontroladores.

ROS

Representa este nombre a las siglas de *Robot Operating System* , que es es uno de los principales entornos de trabajo para el desarrollo de software del ámbito de la robótica móvil.

Los comienzos de la historia de ROS se remontan a un proyecto de la Universidad de Standford en 2006 desarrollado por Keenan Wyrobek y Eric Berger bajo el nombre de Switchyard en el Laboratorio de Inteligencia Artificial para optimizar el tiempo dedicado en el desarrollo de programas orientados a la robótica. Para ello se generaba una red de procesos que se comunicasen entre ellos simultáneamente.

La colaboración de individuos y grupos de desarrollo de todo el planeta lo ha hecho transformarse en un entorno robusto y flexible para la creación y uso de software en robótica.

Este entorno y librería de software es una de las principales causas para el rápido desarrollo de la robótica móvil en los últimos tiempos, ya que gracias a su modularidad ha permitido el desarrollo de partes individuales del software de un robot. De esta manera se ha hecho asequible para una persona interesada la comprensión y diseño de robots funcionales el uso de los mismos.

ROS es un perfecto ejemplo de como ha resultado fundamental mantener la filosofía de Linux de creación de software libre y modular por parte de usuarios de todo el planeta para alcanzar la magnitud que tiene este proyecto hoy en día.

A.2.4. Elementos empleados en la programación de algoritmos

En este apartado vamos a mencionar específicamente los principales componentes usados a la hora de generar soluciones informáticas para lograr los objetivos del trabajo. Estos elementos van a ser divididos en dos grandes subgrupos.

Software

A pesar de que todas las soluciones informáticas utilizadas son muestras de software, en esta sección se hace referencia tanto a los programas como a los medios que han servido como herramientas de desarrollo de software.

- **VirtualBox**

  VirtualBox es una aplicación que permite la generación de máquinas virtuales que contengan sistemas operativos como Linux, de ahí su aplicación. Desde 2007 cuenta con una licencia pública general de GNU que garantiza al usuario final del software la libertad de usar, estudiar compartir y de modificar el software [28].

- **CMake**

  CMake es una herramienta de código abierto, multi-plataforma de generación o automatización de código, programada en C y C++. En este trabajo ha sido empleado como la herramienta empleaada por el compilador para construir los archivos nativos para el sistema operativo empleado.

- **Stage**

  Stage es un a aplicación de Ubuntu que proporciona *framework* para controlar y simular robots y sensores. Proporciona un marco virtual con robots móviles y sus sensores, así como el entorno con el que interactúan dichos robots. [29]

- **Virtual Studio Code**

  Virtual Studio Code, o también comunmente denominado VS Code, es una aplicación multi-plataforma que no solo sirve como editor de texto en el que escribir código, si no como estructura a la que añadir numereosas

- **Docker**

  Este proyecto de software libre y código abierto fue comenzado por su creador S. Hykes en marzo de 2013 con la finalidad de automatizar el despliegue de aplicaciones dentro de contenedores de software [30]. De esta forma he podido verificar el funcionamiento de los códigos desarrollados dentro de VS Code, que es la plataforma que he empleado para implementar el código que ha sido desarrollado.

- **GitHub**

  GitHub es una interfaz online basada en Git, una plataforma de código abierto de control de versiones de software, que permite a diferentes personas acceder e incluso editar diferentes partes de un programa en tiempo real de forma simultanea. Por ello es la principal plataforma de desarrollo conjunto de software hoy en día. [5]

Librerías de C++

Una de los principales potenciales de C++ como lenguaje de programación es la posibilidad de importar en tu proyecto librerías, es decir, un conjunto de elementos y de funciones previamente generadas por otros programadores. Esto puede ser de gran ayuda, ya que debido a que C++ se puede considerar un lenguaje de programación veterano comparado con muchos otros como Python, y por lo tanto probablemente alguien se haya tenido que enfrentar a una problemática similar a la del programa que te concierne. Por esta razón es fácil encontrar soluciones que se ajusten a las necesidades de cada programa.

Las librerías que han sido empleadas a lo largo del proyecto, y que por lo tanto se ha requerido aprender a emplearlas han sido las siguientes.

- OpenCV

  OpenCV es una librería para C++ y Python de código abierto, aunque inicialmente fue desarrollada por Intel y lanzada en junio del año 2000. Como se puede intuir a partir de su nombre es de gran ayuda para numerosas aplicaciones de visión por computador Un empleo de ello puede ser el análisis y división de imágenes, principal contenido de este trabajo.

- Boost

  Boost hace referencia a un conjunto de bibliotecas de software libre creadas para extender las funcionalidades del lenguaje de programación C++. En este proyecto ha sido empleado en para poder trabajar con los programas de generación de rutas creados por I. Carrizo.

- Chrono

  Se ha empleado esta librería a la hora de obtener medidas precisas de tiempos de procesado de los diferentes programas implementados. Permite una precisión de picosegundos

A.3. Fragmentos de código mencionados

```
class Matrix
{
public:
    int rows;//numero de filas
    int cols;//numero de columnas
    int maxVal;//maximo color en la escala de grises
    string format;//permite guardar formato del archivo
    string comment;//permite guardar comentarios de los datos
    int *arr;//puntero al primer elemento del vector que almacene los
   valores grises de la imagen

    bool cargapgm(string fileName); //lee y carga la imagen pgm
    void binariza(int umbral);      //binariza los colores de la imagen
   para facilitar su procesado
};
```

Fragmento de código A.1: Clase para representar la matriz que contiene la imagen y otros valores que se encuentran en la misma.

```
class Celda
{
public:
    int identificador;//identifica cada celda numericamente
    int minF;//minimo pixel (fila)
    int maxF;//maximo pixel (fila)
    int minC;//minimo pixel (columna)
    int maxC;//maximo pixel (columna)
    int minDim;//dimension minima de la celda
    bool ocupado;//es true si la celda contiene un pixel negro (255)
    vector<CelAdy> ady;// contiene las referencias de las celdas
   adyacentes de la lista de adyacentes

    void checkocupada(MatrixImage mat);//verifica si la celda esta
   ocupada y da valor al parametro 'ocupado', que en principio sera
   false
    void minimadimension();//guarda en minDim la minima dimension que
   sera comparada con la precision
    Segmento segmentoHorizComun(Celda c);// devuelve el segmento
   horizontal en comun que tiene con otra celda
    Segmento segmentoVertiComun(Celda c);// devuelve el segmento
   vertical en comun que tiene con otra celda
    void adyacentes(vector<Celda> listaCell);//escribe por archivo el
   numero de puntos que coinciden con cada celda adyacente
};
```

Fragmento de código A.2: Clase que contendrá toda la información de cada celda.

```
void Matrix::binariza(int umbral)
{
    for (size_t i = 0; i < rows; i++)
    {
        for (size_t j = 0; j < cols; j++)
        {
            if (arr[j + i * cols] < umbral)
                arr[j + i * cols] = 0;
            else
                arr[j + i * cols] = 255;
        };
    };
    return;
}
```

Fragmento de código A.3: Función que binariza el valor de los píxeles que conforman la imagen y así facilitar su procesado.

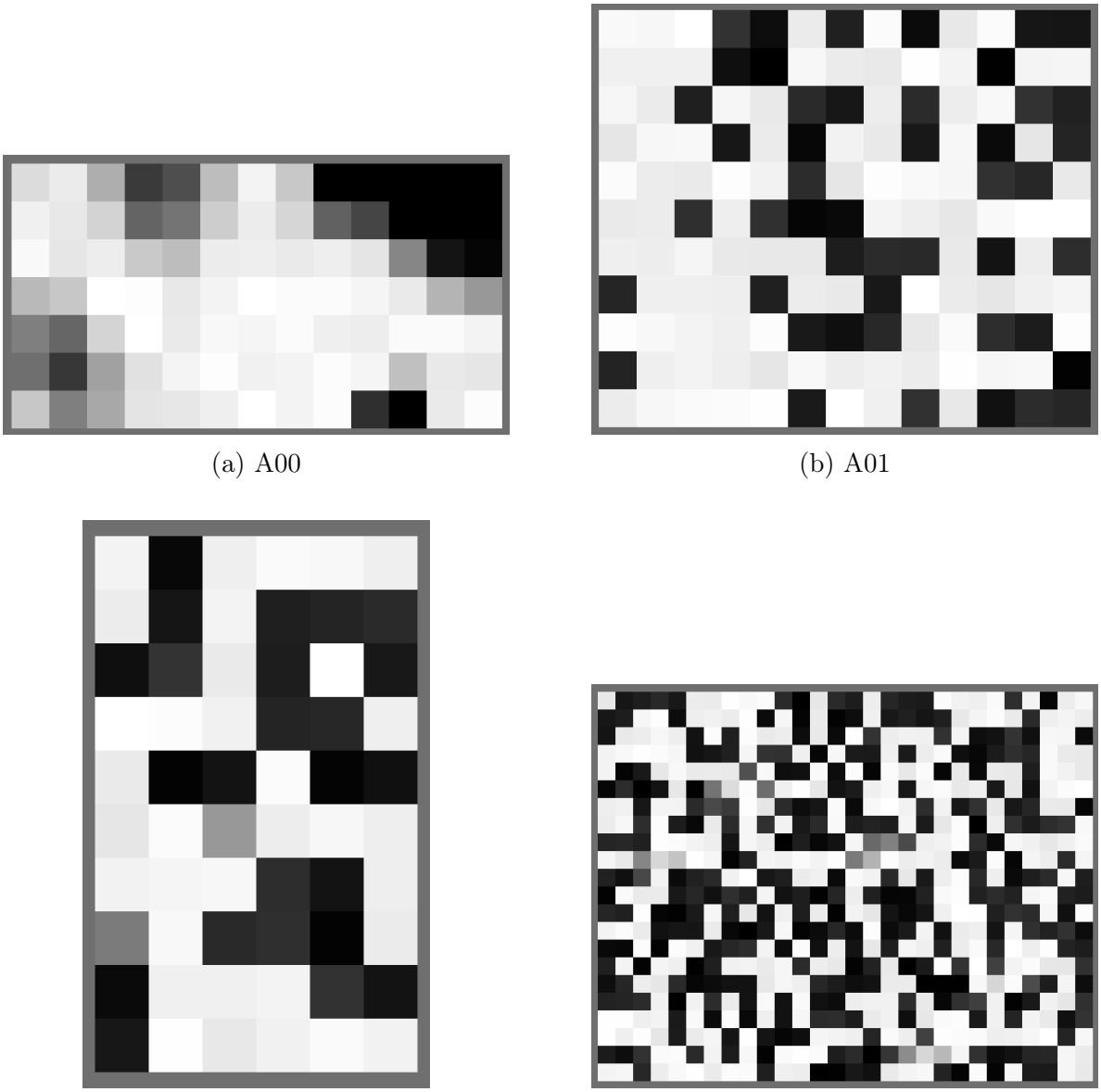
```
void escribirArchivo(vector<Celda> listaCeldas)
{
    string filename("grafo.txt");
    fstream file_out;
    //abre el archivo para escribir en el eliminando el contenido
   anterior en caso de haberlo
    file_out.open(filename, ios_base::out | ios_base::trunc);

    if (!file_out.is_open())
    {
        cout << "failed to open " << filename << '\n';
    }
    else
    {
        file_out << listaCeldas.size() << endl; //numero de celdas
        for (auto cel : listaCeldas)
        {
            file_out << cel.identificador << ";";
            if (cel.ocupado)
                file_out << 1 << ";";
            else
                file_out << 0 << ";";
            file_out << 4 << ";"; //numero de vertices
            //en los algoritmos implementados es siempre igual a 4
            file_out << cel.minF << "," << cel.minF << ","
```

```
                        << cel.maxF << "," << cel.maxF << ";";
            cel.adyacentes(listaCeldas);
            file_out << cel.ady.size() << ";";
            for (auto l : cel.ady)
            {
                file_out << l.ident << "(2):("
                         << l.seg.inicio.x << "," << l.seg.inicio.y
                         << ")("
                         << l.seg.final.x << "," << l.seg.final.y
                         << ");";
            }
            file_out << endl; //salto de linea
        }
        file_out.close(); //cierro el archivo
    }
}
```

Fragmento de código A.4: Función que construye el grafo en un archivo de texto con un formato preestablecido a partir de un vector que contenga las celdas.

A.4. Ejemplos de los mapas generados

Se muestran en este Anexo ejemplos de los mapas generados de cada uno de los grupos de estudio mencionados en la Sección 4.1.2.

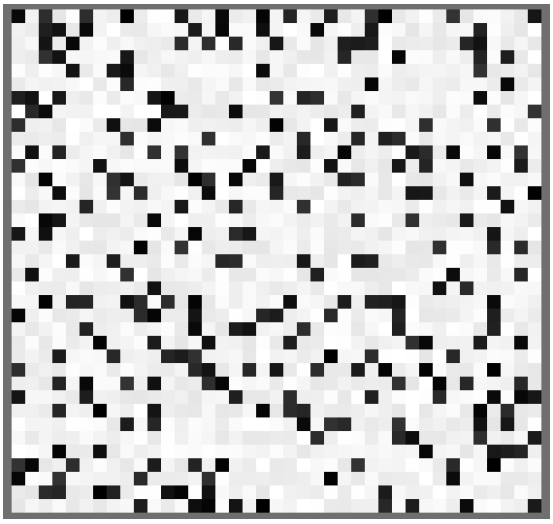


(a) A00 (b) A01

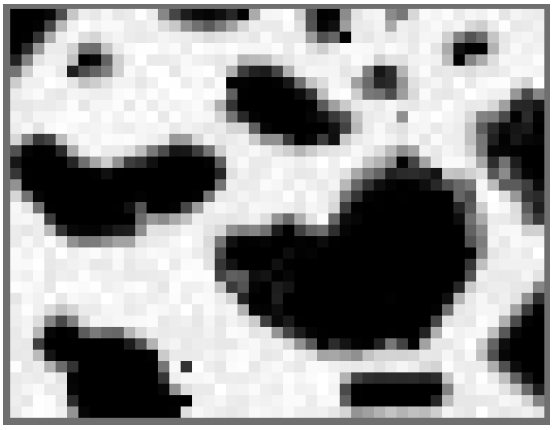
(c) A10 (d) A11

Figura A.6: Ejemplos escalados de mapas pequeños.

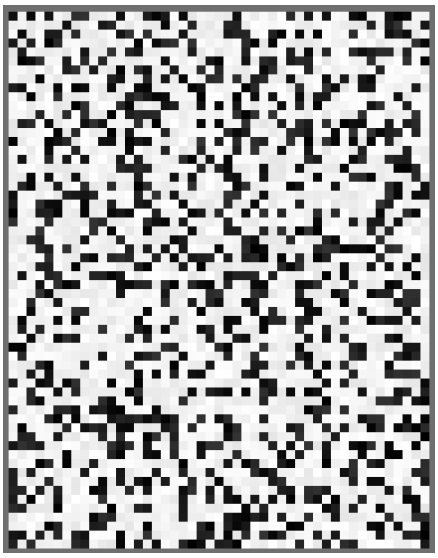
(a) B00



(b) B01

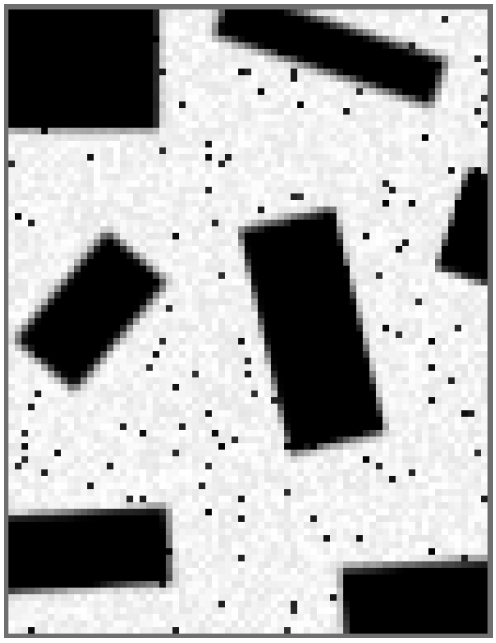


(c) B10

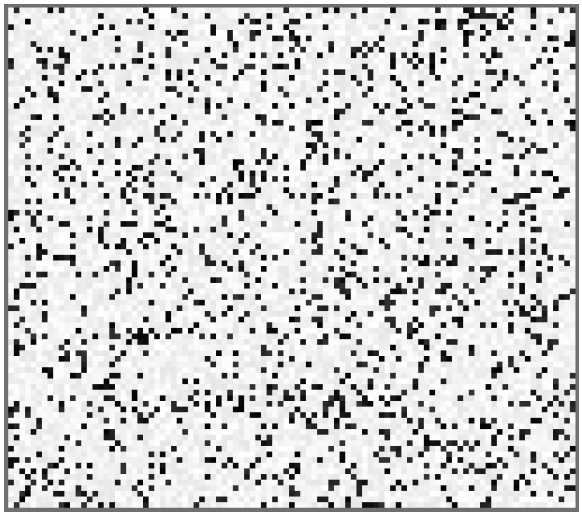


(d) B11

Figura A.7: Ejemplos escalados de mapas medianos.

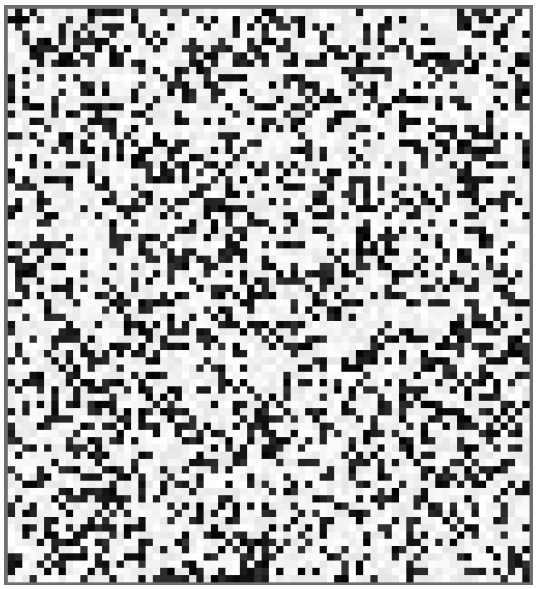


(a) C00



(b) C01

(c) C10



(d) C11

Figura A.8: Ejemplos escalados de mapas grandes.

A.5. Contenido del grafo

Tal y como se expresa en la Sección 3.4, cada una de las líneas del documento de texto tiene una codificación preestablecida. Cada una de las siguientes partes de la definición de una celda queda separada de la siguiente por un '**;**'. Cada una de las líneas representará una de las celdas que compone el grafo. Su número de celdas quedará expresado en la primera de las líneas que componen el documento de texto.Las diferentes características expresadas son:

- **identificador**: cifra que caracterizará a la celda y la distinguirá de las otras.
- **ocupado**: su valor será 1 si la celda está ocupada y 0 en el caso opuesto.
- **numero de vértices**: número de vértices de la celda
- **num_x**: número de coordenadas X de los vértices.
- **lista_x**:conjunto de coordenadas de X, numerado en orden horario comenzando por el elemento superior derecho.
- **num_y**: número de coordenadas Y de los vértices.
- **lista_y**: conjunto de coordenadas de Y, numerado en orden horario comenzando por el elemento superior derecho.
- **numero de celdas adyacentes**: número de celdas que comparten al menos una cara con la la celda en cuestión.
- **identificador celda adyacente(puntos en común):(primer punto)...(ultimo punto)**: la primera cifra representará el identificador de la celda adyacente (habrá un numero de celdas adyacentes que ya se ha indicado por el anterior dígito antes del punto y coma. Separado por paréntesis a continuación del indicador de la celda aparecerá el número de puntos que tienen en común. Tras cerrar los paréntesis y el carácter ':' aparecerán los n puntos en común que tengan las celdas,vi endosé cada uno representado por dos cifras separadas por una coma entre paréntesis. La primera de ella representará las columnas (X) mientras que la segunda serán las columnas (Y).

```
225
1;1;4;4;0,1,0,1;4;0,0,1,1;2;  2(2):(1,0)(1,1);  16(2):(0,1)(1,1);
2;1;4;4;2,3,2,3;4;0,0,1,1;3;  1(2):(2,0)(2,1);  3(2):(3,0)(3,1);  17(2):(2,1)(3,1);
3;0;4;4;4,5,4,5;4;0,0,1,1;3;  2(2):(4,0)(4,1);  4(2):(5,0)(5,1);  18(2):(4,1)(5,1);
4;0;4;4;6,7,6,7;4;0,0,1,1;3;  3(2):(6,0)(6,1);  5(2):(7,0)(7,1);  19(2):(6,1)(7,1);
5;0;4;4;8,9,8,9;4;0,0,1,1;3;  4(2):(8,0)(8,1);  6(2):(9,0)(9,1);  20(2):(8,1)(9,1);
6;0;4;4;10,11,10,11;4;0,0,1,1;3;  5(2):(10,0)(10,1);  7(2):(11,0)(11,1);  21(2):(10,1)(11,1);
7;1;4;4;12,13,12,13;4;0,0,1,1;3;  6(2):(12,0)(12,1);  8(2):(13,0)(13,1);  22(2):(12,1)(13,1);
8;1;4;4;14,15,14,15;4;0,0,1,1;3;  7(2):(14,0)(14,1);  9(2):(15,0)(15,1);  23(2):(14,1)(15,1);
9;0;4;4;16,17,16,17;4;0,0,1,1;3;  8(2):(16,0)(16,1);  10(2):(17,0)(17,1);  24(2):(16,1)(17,1);
10;0;4;4;18,19,18,19;4;0,0,1,1;3;  9(2):(18,0)(18,1);  11(2):(19,0)(19,1);  25(2):(18,1)(19,1);
11;1;4;4;20,21,20,21;4;0,0,1,1;3;  10(2):(20,0)(20,1);  12(2):(21,0)(21,1);  26(2):(20,1)(21,1);
12;1;4;4;22,23,22,23;4;0,0,1,1;3;  11(2):(22,0)(22,1);  13(2):(23,0)(23,1);  27(2):(22,1)(23,1);
13;1;4;4;24,25,24,25;4;0,0,1,1;3;  12(2):(24,0)(24,1);  14(2):(25,0)(25,1);  28(2):(24,1)(25,1);
14;1;4;4;26,27,26,27;4;0,0,1,1;3;  13(2):(26,0)(26,1);  15(2):(27,0)(27,1);  29(2):(26,1)(27,1);
15;1;4;4;28,29,28,29;4;0,0,1,1;2;  14(2):(28,0)(28,1);  30(2):(28,1)(29,1);
```

Fragmento de código A.5: Fragmento del grafo de la descomposición en retícula de un Mapa A10. (Líneas 0-15)

A.6. Algoritmos de comparación entre los diferentes métodos implementados

Generador de mapas

Basándose en números aleatorios delimitados, se genera un array de valores que representan las gamas de gris que componen una imagen `.pgm`. Se imponen determinadas restricciones en las adyacencia de los obstáculos entre si para cumplir las restricciones relevantes al grado de aislamiento de los obstáculos entre si.

```
#include <ctime>
// Nueva semilla de randomizar a partir del procesador
rand((unsigned)time(0));
time_t now = time(0);
// Generacion de numero aleatorio
random_number = 68 + rand() % 34;
```

Fragmento de código A.6: Generación de números aleatorios enteros en un rango entre 68 y 100 ambos inclusive.

Timer

Para la implelmentación de un método de medición de tiempos se ha generado a partir del uso de una librería de medición de los tiempos internos de procesos del ordenador. La funcionalidad de la librería `chrono` escogida para la medición entrega los resultados en picosegundos (μs). A pesar de ello posteriormente los resultados en las tablas del Anexo A.7 se muestren en microsegundos (ms) para mayor claridad.

```
#include <chrono>

class Timer
{
public:
    Timer() {m_StartTimepoint = chrono::high_resolution_clock::now();}
    ~Timer(){Stop();}

    void Stop()
    {   (...)
        auto duration = stop - start;
        double ms = duration * 0.001; // duracion en microsegundos
    }   (...)
};
```

Fragmento de código A.7: Fragmentos de la clase cuya aplicación en el código original permite medir tempos de procesado.

A.7. Resultados de las comparaciones realizadas

Se muestran en este anexo los resultados de las simulaciones realizadas. Para representar el valor medio y la desviación típica se emplean las siguientes letras griegas:

- **Valor medio -** μ
- **Desviación típica -** σ

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	23,577	83,593	4	0,693	0,601
2	22,88	81,919	4	0,846	851
3	14,755	64,14	4	0,602	0,558
4	14,205	59,095	4	0,967	0,801
5	26,768	93,305	4	0,659	1
6	24,414	88,503	4	0,97	0,552
7	22,874	72,454	4	0,539	0,659
8	18,26	68,625	4	0,953	0,551
9	18,659	70,739	4	0,510	0,707
10	35,686	109,259	4	0,633	0,854
μ	22,208	79,163	4	0,744	0,708
σ	5,982	14,407	0	0,181	0,152
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	23,443	36,528	9,255	0,774	0,715
2	16,097	44,096	12,044	0,847	0,611
3	23,827	47,211	9,03	0,676	0,815
4	24,925	33,004	7,382	0,976	0,722
5	17,329	42,147	11,572	0,756	0,537
6	35,072	31,633	4,722	0,874	0,687
7	19,489	41,388	11,165	0,554	0,88
8	23,847	37,579	8,095	0,854	0,897
9	20,504	40,017	9,994	0,42	0,226
10	31,26	35,778	6,58	0,750	1
μ	23,579	38,938	8,984	0,743	0,709
σ	5,606	4,677	2,219	0,155	0,225

Tabla A.1: Resultados de los ensayos de aplicación de descomposiciones en Mapas A00.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	31,15	81,598	4	0,944	0,344
2	22,207	97,684	4	0,958	0,647
3	17,892	110,694	4	0,848	0,788
4	18,035	98,893	4	0,713	0,609
5	21,988	47,354	4	0,647	0,327
6	26,749	114,177	4	0,565	0,707
7	23,305	61,666	4	0,921	0,49
8	31,982	77,55	4	0,475	0,618
9	14,249	82,383	4	0,454	0,788
10	28,236	85,061	4	0,706	1
μ	23,579	85,706	4	0,723	0,591
σ	5,606	19,738	0	0,18	0,207
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	65,782	51,975	5,274	0,950	0,726
2	64,615	53,948	5,899	1,021	0,748
3	62,693	41,398	6,968	0,803	0,796
4	64,909	37,583	5,679	0,685	0,286
5	61,347	36,751	7,616	0,655	0,836
6	79,756	44,085	3,855	0,538	0,380
7	66,573	35,870	4,640	0,892	0,239
8	58,870	38,130	7,664	0,532	0,702
9	51,912	43,098	8,160	0,426	0,684
10	51,693	45,635	10,397	0,666	0,322
μ	62,815	42,847	6,615	0,717	0,572
σ	7,602	5,948	1,834	0,185	0,223

Tabla A.2: Resultados de los ensayos de aplicación de descomposiciones en Mapas A01.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	27,570	91,676	4	0,932	0,854
2	22,726	94,413	4	0,508	0,675
3	19,104	68,311	4	0,796	0,571
4	21,594	119,555	4	0,646	1,000
5	28,665	76,457	4	0,894	0,630
6	31,154	100,877	4	0,877	0,460
7	22,219	91,639	4	0,915	0,524
8	11,505	37,550	4	0,375	0,500
9	30,172	125,924	4	0,915	0,645
10	23,991	104,108	4	0,907	0,504
μ	23,670	91,051	4	0,776	0,636
σ	6,061	24,290	0	0,188	0,172
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	43,614	44,117	8,961	0,913	1,000
2	47,622	37,082	5,889	0,414	0,735
3	45,007	32,709	7,160	0,480	0,576
4	44,538	56,860	8,330	0,470	0,854
5	49,014	55,390	5,502	0,538	0,707
6	54,555	59,096	2,468	0,529	0,185
7	51,270	59,180	4,556	0,647	0,825
8	49,891	32,972	4,713	0,277	0,707
9	47,525	38,546	6,451	0,729	0,657
10	47,298	45,997	7,036	0,633	0,500
μ	48,033	46,195	6,107	0,563	0,694
σ	3,151	10,178	1,822	0,168	0,227

Tabla A.3: Resultados de los ensayos de aplicación de descomposiciones en Mapas A10.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	28,364	41,550	4	0,525	0,662
2	27,216	114,272	4	0,787	0,273
3	17,488	126,712	4	0,822	0,724
4	30,162	62,078	4	0,550	0,750
5	22,484	63,615	4	0,872	0,527
6	12,475	84,642	4	0,882	0,587
7	22,479	116,197	4	0,377	0,419
8	25,081	108,682	4	0,683	0,727
9	25,069	128,729	4	0,917	0,503
10	29,230	104,864	4	0,904	0,843
μ	23,705	95,134	4	0,732	0,601
σ	5,951	28,777	0	0,179	0,165
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	92,365	52,221	4,511	0,452	0,250
2	93,371	43,734	3,493	0,673	0,490
3	55,821	32,299	8,576	0,706	0,250
4	39,414	43,152	10,051	0,476	1,000
5	82,872	45,753	5,302	0,728	0,590
6	81,778	59,039	5,590	0,739	0,565
7	87,268	52,797	4,829	0,333	0,667
8	104,105	44,450	1,725	0,632	0,500
9	59,726	44,596	6,664	0,977	0,707
10	56,590	72,275	7,399	0,794	0,564
μ	75,331	49,032	5,814	0,661	0,558
σ	19,874	10,290	2,322	0,208	0,218

Tabla A.4: Resultados de los ensayos de aplicación de descomposiciones en Mapas A11.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	32,759	163,147	4	0,560	0,682
2	32,126	177,074	4	0,834	0,671
3	44,230	138,112	4	0,876	0,884
4	45,342	135,161	4	0,886	0,829
5	54,179	118,036	4	0,583	0,730
6	42,555	113,658	4	0,680	0,613
7	45,541	138,766	4	0,862	0,799
8	30,804	129,593	4	0,796	0,635
9	27,887	135,586	4	0,664	0,514
10	43,342	91,789	4	0,647	0,803
μ	39,877	134,092	4	0,739	0,716
σ	8,009	22,843	0	0,119	0,108
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	53,275	65,731	13,962	0,524	0,997
2	95,109	67,932	8,741	0,798	0,888
3	76,029	73,646	10,268	0,813	0,506
4	85,101	48,015	10,126	0,895	0,678
5	67,410	77,512	11,842	0,638	0,889
6	59,644	75,714	13,759	0,786	0,679
7	73,371	59,801	10,762	0,808	0,417
8	73,580	68,089	10,416	0,793	0,715
9	88,303	79,345	10,060	0,740	0,397
10	43,535	64,416	18,837	0,732	0,858
μ	71,536	68,020	11,877	0,753	0,702
σ	15,246	8,917	2,802	0,099	0,199

Tabla A.5: Resultados de los ensayos de aplicación de descomposiciones en Mapas B00.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	39,815	176,200	4	0,392	0,569
2	30,789	174,177	4	0,673	0,569
3	44,270	138,482	4	0,676	0,585
4	37,700	150,244	4	0,927	0,549
5	41,861	96,422	4	0,703	0,893
6	44,068	181,904	4	0,677	0,637
7	54,573	138,030	4	0,728	0,672
8	31,196	140,586	4	0,655	0,787
9	47,583	181,938	4	0,810	0,779
10	30,415	155,606	4	0,807	0,848
μ	40,227	153,359	4	0,705	0,689
σ	7,531	25,422	0	0,132	0,121
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	218,274	119,451	3,934	0,530	0,531
2	210,346	134,452	4,517	0,605	0,636
3	189,832	136,880	8,352	0,647	0,669
4	179,989	124,804	8,823	0,964	0,383
5	206,839	131,007	7,092	0,689	0,779
6	176,956	116,783	10,483	0,663	0,475
7	174,640	137,678	12,753	0,700	1,000
8	191,362	136,560	8,070	0,546	0,387
9	194,994	131,772	7,871	0,819	0,386
10	174,088	141,924	15,563	0,762	0,850
μ	191,732	131,131	8,746	0,692	0,617
σ	15,011	7,846	3,327	0,124	0,219

Tabla A.6: Resultados de los ensayos de aplicación de descomposiciones en Mapas B01.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	35,284	143,570	4	0,679	0,930
2	52,362	141,417	4	0,628	0,620
3	32,180	88,959	4	0,865	0,546
4	47,745	174,401	4	0,782	0,693
5	34,746	150,752	4	0,546	0,468
6	35,293	136,357	4	0,801	0,574
7	52,498	190,730	4	0,722	1,000
8	38,238	181,815	4	0,550	0,887
9	51,214	146,529	4	0,872	0,570
10	49,589	176,993	4	0,385	0,578
μ	42,915	153,152	4	0,683	0,686
σ	7,988	28,170	0	0,157	0,185
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	94,170	73,235	3,334	0,611	0,981
2	92,700	82,968	6,167	0,577	0,801
3	80,218	80,428	7,941	0,826	0,530
4	71,847	84,456	11,781	0,702	0,703
5	72,526	64,263	10,473	0,536	0,562
6	86,709	74,403	7,634	0,709	0,435
7	80,979	75,282	7,732	0,693	0,805
8	77,681	65,814	8,503	0,565	0,723
9	75,258	88,887	9,487	0,892	0,759
10	76,033	65,496	8,889	0,458	0,532
μ	80,812	75,523	8,194	0,686	0,683
σ	7,533	8,161	2,205	0,185	0,157

Tabla A.7: Resultados de los ensayos de aplicación de descomposiciones en Mapas B10.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	32,214	171,017	4	0,850	0,408
2	33,559	146,690	4	0,770	0,925
3	46,403	124,732	4	0,679	0,611
4	45,438	143,771	4	0,484	0,539
5	52,234	191,991	4	0,784	0,733
6	44,656	148,751	4	0,707	0,739
7	29,026	127,213	4	0,815	0,447
8	49,455	124,194	4	0,623	0,718
9	61,173	209,271	4	0,957	0,648
10	47,707	196,996	4	0,729	0,751
μ	44,187	158,463	4	0,740	0,652
σ	9,414	30,142	0	0,123	0,148
Descomposición ectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	150,806	45,523	11,536	0,829	0,776
2	207,386	74,581	5,273	0,686	0,797
3	211,608	134,493	5,099	0,591	0,859
4	214,502	98,317	3,072	0,427	0,624
5	132,114	71,281	11,684	0,693	0,471
6	192,046	83,027	6,613	0,647	0,134
7	193,690	85,106	6,579	0,720	0,211
8	163,670	85,384	9,415	0,500	0,792
9	186,413	84,572	7,153	0,926	0,424
10	200,039	84,665	6,293	0,662	0,436
μ	185,227	84,695	7,272	0,668	0,552
σ	26,179	21,179	2,655	0,137	0,244

Tabla A.8: Resultados de los ensayos de aplicación de descomposiciones en Mapas B11.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	86,584	224,251	4	0,733	0,595
2	63,181	101,821	4	0,830	0,849
3	85,695	201,617	4	0,854	0,824
4	74,995	176,196	4	0,703	0,721
5	82,703	187,369	4	0,767	0,721
6	84,446	153,463	4	0,755	0,873
7	84,733	183,729	4	0,660	0,822
8	77,279	177,029	4	0,814	0,644
9	49,192	154,402	4	0,609	0,887
10	75,308	164,628	4	0,648	0,611
μ	76,412	172,450	4	0,737	0,755
σ	11,327	31,072	0	0,078	0,105
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	159,039	71,525	14,485	0,751	1,000
2	188,406	106,458	11,514	0,868	0,619
3	176,434	88,514	13,038	0,899	0,683
4	136,931	85,591	16,684	0,736	0,586
5	111,279	58,898	26,252	0,810	0,651
6	144,724	96,691	16,319	0,801	0,411
7	167,015	77,274	14,009	0,730	0,690
8	135,189	74,619	20,508	0,865	0,891
9	122,173	70,382	21,223	0,706	0,795
10	166,695	66,331	14,379	0,724	1,000
μ	150,788	79,628	16,841	0,789	0,743
σ	23,472	13,834	4,289	0,066	0,196

Tabla A.9: Resultados de los ensayos de aplicación de descomposiciones en Mapas C00.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	52,479	226,058	4	0,580	0,635
2	91,716	221,476	4	0,777	0,654
3	89,282	232,263	4	0,729	0,627
4	81,803	188,399	4	0,845	0,833
5	70,184	144,629	4	0,700	0,615
6	60,742	76,706	4	0,719	0,620
7	79,943	178,035	4	0,792	0,934
8	73,172	190,504	4	0,712	0,542
9	72,229	216,049	4	0,728	0,724
10	79,897	173,459	4	0,574	0,707
μ	75,145	184,758	4	0,716	0,696
σ	11,500	44,491	0	0,081	0,113
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	226,092	154,377	11,882	0,676	1,000
2	208,800	173,910	13,013	0,805	0,492
3	231,477	142,801	10,846	0,788	0,385
4	195,314	136,560	14,006	0,872	0,586
5	248,249	164,634	4,165	0,690	0,691
6	229,068	136,431	11,497	0,777	0,664
7	145,184	142,202	16,454	0,812	0,860
8	167,753	145,859	16,261	0,712	0,763
9	203,927	139,760	13,099	0,783	0,665
10	210,223	145,947	12,785	0,581	0,390
μ	206,609	148,248	12,401	0,750	0,659
σ	29,448	11,790	3,260	0,080	0,205

Tabla A.10: Resultados de los ensayos de aplicación de descomposiciones en Mapas C01.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	64,770	199,457	4	0,825	0,573
2	88,352	193,067	4	0,686	0,523
3	83,990	162,516	4	0,863	0,813
4	68,120	204,119	4	0,719	0,921
5	72,077	211,772	4	0,844	0,575
6	80,530	138,182	4	0,683	0,788
7	95,762	276,869	4	0,923	0,800
8	59,000	169,213	4	0,792	0,864
9	75,830	188,831	4	0,838	0,696
10	61,352	172,018	4	0,725	0,794
μ	74,978	191,604	4	0,790	0,735
σ	11,507	35,341	0	0,078	0,129
Descomposición rectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	193,404	109,370	9,529	0,670	0,561
2	191,679	113,200	10,851	0,564	0,511
3	138,953	140,508	14,150	0,791	0,653
4	183,031	101,847	12,058	0,574	0,280
5	170,437	111,932	13,663	0,724	0,664
6	209,092	105,698	3,616	0,547	0,711
7	196,656	97,928	9,291	0,811	0,876
8	117,539	108,677	15,300	0,640	0,196
9	178,646	98,934	12,714	0,713	0,304
10	178,527	99,964	13,636	0,635	0,920
μ	175,797	108,806	11,481	0,667	0,568
σ	26,370	11,775	3,222	0,088	0,235

Tabla A.11: Resultados de los ensayos de aplicación de descomposiciones en Mapas C10.

Descomposición en retícula					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	81,113	273,456	4	0,735	0,894
2	60,345	211,694	4	0,641	0,794
3	66,574	94,148	4	0,761	0,518
4	81,173	144,388	4	0,656	0,789
5	63,161	239,520	4	0,788	0,743
6	52,548	213,439	4	0,888	0,650
7	94,576	197,859	4	0,782	0,545
8	71,508	230,907	4	0,807	0,801
9	74,533	173,957	4	0,627	0,499
10	98,194	187,137	4	0,698	0,608
μ	74,373	196,650	4	0,738	0,684
σ	13,911	48,233	0	0,079	0,131
Descomposición ectangular					
Número de ensayo	t_{desc} (ms)	t_{ruta} (ms)	S_{celda}	A_{ocup}	A_{ruta}
1	277,126	46,258	7,625	0,662	0,255
2	232,113	145,453	10,234	0,552	0,827
3	154,263	142,452	16,012	0,681	0,475
4	211,735	75,751	14,736	0,594	0,710
5	235,396	81,723	9,290	0,728	0,953
6	231,950	178,714	10,692	0,828	0,133
7	218,595	133,881	12,744	0,714	0,523
8	285,223	165,373	5,921	0,755	0,707
9	239,764	93,890	8,589	0,544	0,532
10	255,306	106,160	7,644	0,610	0,573
μ	234,147	116,966	10,349	0,667	0,568
σ	34,744	40,611	3,094	0,088	0,235

Tabla A.12: Resultados de los ensayos de aplicación de descomposiciones en Mapas C11.

A.8. Fundamentos teóricos de los métodos de partición en celdas y del trazado de rutas

A.8.1. Triangulación de Delaunay

La triangulación de Delaunay, denominada en honor del matemático ruso Borís Nikolaevich Delone que la ideó en 1934, es una red de triángulos conexa y convexa que cumple la condición de Delaunay.[31] Esta condición dice que la circunferencia circunscrita de cada triángulo de la red no debe contener ningún vértice de otro triángulo. Las triangulaciones de Delaunay tienen importante relevancia en el campo de la geometría computacional, especialmente en gráficos 3D por computadora.

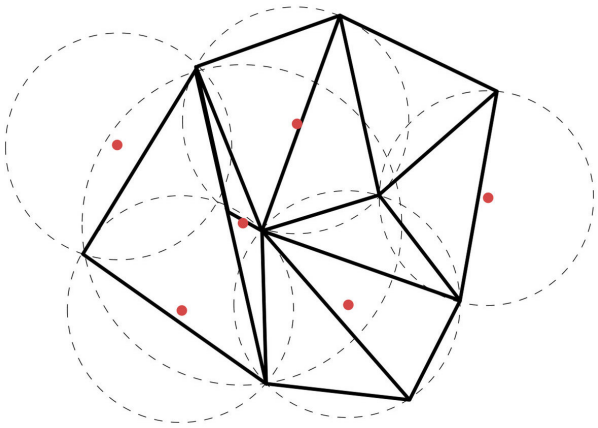


Figura A.9: Triangulación de Delaunay con las circunferencias circunscritas a los triángulos, siendo marcados sus centros en rojo.

Extraído del artículo *"LIDAR-based Collision-Free Space Estimation Approach"* de Unger, Miklós and Horváth,2020 [32]

A.8.2. Algoritmo de Djikstra

El procedimiento teórico detrás de este algoritmo es examinar todas las rutas más cortas que comienzan desde el nodo de origen y conducen a todos los demás nodos. El algoritmo finaliza cuando se obtiene el camino más corto desde el vértice original hasta los vértices restantes que componen el gráfico, quedando dicho valor asignado a ellos.

Bibliografía

[1] R. Mariani, "An overview of autonomous vehicles safety," in *2018 IEEE International Reliability Physics Symposium (IRPS)*, 03 2018, pp. 6A.1–1–6A.1–6, ISSN: 1938-1891.

[2] N. Adzhar, Y. Yusof, and M. A. Ahmad, "A Review on Autonomous Mobile Robot Path Planning Algorithms," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 3, pp. 236–240, 2020.

[3] J. A. Fernández Madrigal, *Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods: Introduction and Methods*. IGI Global, Sep. 2012, google-Books-ID: JbaeBQAAQBAJ.

[4] C. K. Wong, J. Schmidt, and W.-K. Yeap, "Using a mobile robot for cognitive mapping." *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2243–2249, 01 2007.

[5] GitHub: Where the world builds software. [Online]. Available: https://github.com/

[6] E. S. Raymond and T. O'Reilly, *The Cathedral and the Bazaar*, 1st ed. USA: O'Reilly; Associates, Inc., 1999.

[7] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots, second edition.* MIT Press, 2011.

[8] C. Mahulea, M. Kloetzer, and R. Gonzalez, *Path Planning of Cooperative Mobile Robots Using Discrete Event Models.* John Wiley & Sons, 2020, google-Books-ID: jqjDDwAAQBAJ.

[9] S. Macenski. (2015) Inflation costmap plugin. [Online]. Available: http://wiki.ros.org/costmap_2d/hydro/inflation

[10] R. Zhou, "Free space detection based on occupancy gridmaps," Master's thesis, Technical University Darmstadt, 2012.

[11] D.-Z. Du and F. Hwang, *Computing in Euclidean Geometry.* WORLD SCIENTIFIC, 1992. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/1657

[12] GKQuadtree | Apple Developer Documentation. [Online]. Available: https://developer.apple.com/documentation/gameplaykit/gkquadtree

[13] GameplayKit programming guide: About GameplayKit. [Online]. Available: https://developer.apple.com/library/archive/documentation/General/Conceptual/GameplayKit_Guide/index.html#//apple_ref/doc/uid/TP40015172

[14] S. Popinet, "Adaptive modelling of long-distance wave propagation and fine-scale flooding during the Tohoku tsunami," *Natural Hazards and Earth System Sciences*, vol. 12, pp. 1213–1227, 2012.

[15] A. C. Olivieri and G. M. Escandar, "Chapter 6 - analytical figures of merit," in *Practical Three-Way Calibration*, A. C. Olivieri and G. M. Escandar, Eds. Boston: Elsevier, 2014, pp. 93–107. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124104082000065

[16] L. Gualtieri, E. Rauch, and R. Vidoni, "Emerging research fields in safety and ergonomics in industrial collaborative robotics: A systematic literature review," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 101998, 02 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S073658452030209X

[17] M. Q. Bakri, A. H. Ismail, M. Hashim, and M. J. A. Safar, "A review on service robots: Mechanical design and localization system," *IOP Conference Series: Materials Science and Engineering*, vol. 705, p. 012003, dec 2019. [Online]. Available: https://doi.org/10.1088/1757-899x/705/1/012003

[18] Unsplash. Collection of free access images. [Online]. Available: https://unsplash.com/collections/1d0vaE8Coz4/tfg

[19] HiSoUR. Sensores robóticos. [Online]. Available: https://www.hisour.com/es/robotic-sensors-42873/

[20] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, 2019. [Online]. Available: https://doi.org/10.1177/1729881419839596

[21] J.-P. P. Laumond, *Robot Motion Planning and Control*. Berlin, Heidelberg: Springer-Verlag, 1998.

[22] A. Baquedano Lasheras, "IMPACTO DE LOS ERRORES DE CALIBRACIÓN EN LA PRECISIÓN DE UN ROBOT," *Proyecto de Fin de Carrera, Universidad de Zaragoza*, 2013.

[23] F. Martín, "Aportaciones a la auto-localización visual de robots autónomos con patas," Ph.D. dissertation, Departamento de Sistemas Telemáticos y Computación Universidad Rey Juan Carlos, 06 2008.

[24] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 249–265, 1987.

[25] B. K. Patle, G. B. L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214914718305130

[26] Coninx, Alex. What are the names of other operating systems for robotics? [Online]. Available: https://www.quora.com/What-are-the-names-of-other-Operating-Systems-for-Robotics

[27] C. M. Lloyd, *Workspace*. New York, NY: Springer New York, 2013, pp. 2356–2356. [Online]. Available: https://doi.org/10.1007/978-1-4419-9863-7_1531

[28] GNU Project - Free Software Foundation. The GNU general public license v3.0. [Online]. Available: https://www.gnu.org/licenses/gpl-3.0.html

[29] (2012) Manual for Stage: The Stage simulator. [Online]. Available: https://codedocs.xyz/CodeFinder2/Stage/md_README.html

[30] S. Hykes. (2013) Docker | App Containerization |. [Online]. Available: https://www.docker.com/

[31] B. Delaunay, "Sur la sphère vide," *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, vol. 1934, no. 6, pp. 793–800, 1934.

[32] M. Unger, E. Horváth, and C. Hajdu, "Lidar-based collision-free space estimation approach," *Hungarian Journal of Industry and Chemistry*, vol. 48, 2020.