UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# Diseño de nuevos algoritmos de guiado y navegación con evasión de colisiones para vehículos aéreos no tripulados

PhD Dissertation


| | |
|---|---|
| Autor | Franklin Samaniego R. |
| Director | Javier Sanchís Saez<br>Sergio García-Nieto |

Instituto Universitario de Automática e Informática Industrial
Departamento de Ingeniería de Sistemas y Automática
Universitat Politècnica de València


15 de noviembre de 2020

Este trabajo ha sido parcialmente financiado por el Gobierno Español a través del Ministerio de Economía y Competitividad bajo el proyecto de Investigación $DPI2015-71443-R$, y por la Administración local de la Generalitat Valenciana a través de los proyectos $GV/2017/029$ y $AICO/2019/055$.

El Autor, ha sido beneficiario de una Beca otorgada por el Instituto de Fomento al Talento Humano (IFTH) ($2015-AR2Q9209$) a través del Gobierno de Ecuador.

# Agradecimientos

Cuatro años han transcurrido desde que inicié el largo camino de trabajo hacia una tesis doctoral, años en los que han existido muchas horas de reuniones, de lectura, de escritura, de programación, etc; donde cada hora superada ha sido un aliciente para continuar con la siguiente. Muchas son las personas que directa o indirectamente, han sido mi apoyo a lo largo de este camino, mencionar a cada uno de ellos es una tarea por demás complicada, no es mi deseo el dejar a ninguno fuera de una lista ni ponerlos en un orden especial.

Deseo extender mi más grande y sincero sentimiento de gratitud y estima hacia mis Tutores: Javier y Sergio, quienes han dedicado, su tiempo, su dedicación y su esfuerzo con el único interés de guiarme hacia la meta de la tesis doctoral. Del mismo modo, quiero agradecer a Raúl que ha sido de gran apoyo durante este trabajo.

A pesar de estar a un océano y a miles de kilómetros de distancia de mi país, la Univeristat Politècnica de València y la ciudad de Valencia me han acogido desde el primer día, es imposible decir que después de haber transcurrido tanto tiempo en España, ahora no sienta a este país como mi segundo hogar. Es muy difícil nombrar a cada una de las personas que han estado a mi alrededor durante esta estancia de estudio y tampoco deseo establecer un orden particular. Así que solo quiero agradecer a todas las personas que forman parte del Departamento de Ingeniería de Sistemas y Automática (DISA) y de manera muy especial a todos y cada uno de mis compañeros de la Sala de Investigación con quienes hemos compartido buenos momentos tanto dentro como fuera de la Universidad.

Por último, pero más importante, quiero agradecer a mi familia en España, mi esposa Glory, dos décadas siendo mi compañera, mi socia, mi cómplice, mi amiga, mi novia, el pilar que me ha sostenido siempre, quien me ha soportado y me ha apoyado a lo largo de este trayecto, mi hermano Diego, mi cuñada Cris y mi amado sobrino, mi ahijado Martín.

*Sinceramente*
***Franklin Eduardo Samaniego Riera***

# Resumen

Debido a la creciente popularidad sobre la variedad de los Vehículos No Tripulados tanto en el campo militar como en el comercial, y de sus capacidades para navegar por diversos entornos, ya sean terrestres, aéreos o marinos, se evidencia que la clásica planificación de trayectorias y movimientos bidimensionales 2D podría no ser suficiente en un futuro inmediato. De esta manera, se debe resaltar que la presente tesis aborda el problema de los Vehículos Aéreos No Tripulados (UAVs) de ala fija. En este sentido, la necesidad de encontrar una trayectoria navegable en el espacio euclídeo 3D se hace cada vez más necesario. En el caso de los UAV, considerar su cinemática para generar trayectorias suaves en tres dimensiones puede tener un interés significativo para la navegación autónoma aérea. Finalmente, los beneficios adicionales que se pueden producir son importantes.

La principal dificultad de este problema es que los vehículos aéreos de características no-holonómicas se ven obligados a avanzar sin la posibilidad de detenerse a través de trayectorias 3D con curvaturas limitadas. En este sentido, se ha investigado la manera de proporcionar una completa caracterización de trayectorias óptimas para UAVs con un radio de giro limitado que se mueve en el plano tridimensional a una velocidad constante.

Para completar tales tareas, un planificador de trayectorias no sólo debe proporcionar rutas tridimensionales para alcanzar una posición de destino sin colisionar con obstáculos, sino también debe asegurar que tal trayectoria sea adecuada para los UAVs que poseen propiedades cinemáticas específicas. Por lo tanto, el desarrollo del trabajo ha completado la algoritmia que genera una

trayectoria discreta tridimensional al definir un conjunto de puntos 3D, resultantes de una división del espacio euclídeo tridimensional de manera dinámica, determinando las mejores opciones de avance, evitando analizar cada espacio del entorno completo. De esta manera, partiendo de los puntos 3D resultantes de la planificación de trayectoria tridimensional, se ha generado una trayectoria en forma de curva suave construida en función de las limitaciones de giro del UAV (resaltando que es difícil asegurar que el camino resultante cumpla con las restricciones cinemáticas en las tres dimensiones simultáneamente). Finalmente, es importante destacar que a menudo las restricciones mencionadas se calculan secuencialmente y de forma bidimensional, sobre un par de dimensiones desacopladas, lo que limita la capacidad de optimización. Para todo ello, se ha desarrollado un algoritmo de suavizado para un planificador de trayectorias que considera las restricciones cinemáticas tridimensionales completas sin desacoplar las dimensiones.

# Abstract

Due to the growing popularity of the variety of Unmanned Vehicles in both the military and commercial fields, and their capabilities to navigate diverse environments, whether land, air or sea, it is evident that the classic two-dimensional 2D trajectory and motion planning may not be enough in the near future. Thus, it should be noted that this thesis addresses the problem of fixed-wing Unmanned Aerial Vehicles (UAVs). In this sense, the need to find a navigable path in 3D Euclidean space becomes more and more necessary. In the case of UAVs, considering their kinematics to generate smooth trajectories in three dimensions may be of significant interest for autonomous air navigation. Finally, the additional benefits that can be produced are important.

The main difficulty of this problem is that air vehicles with non-holonomic characteristics are forced to advance without the possibility of stopping through 3D trajectories with limited curvatures. In this regard, research has been conducted to provide a complete characterization of optimal trajectories for UAVs with a limited turning radius that move in the 3D plane at a constant speed.

To complete such tasks, a path planner must not only provide three-dimensional paths to reach a target position without colliding with obstacles, but must also ensure that such a path is suitable for UAVs that possess specific kinematic properties. Therefore, the development of the work has completed the algorithm that generates a discrete three-dimensional path by defining a set of 3D points, resulting from a division of the three-dimensional Euclidean space in a dynamic way, determining the best forward options, avoiding to analyze each space of the whole environment. In this way, starting from the 3D points

resulting from the three-dimensional path planning, a smooth curve path has been generated, built according to the UAV turning constraints (highlighting that it is difficult to ensure that the resulting path meets the kinematic constraints in the three dimensions simultaneously). Finally, it is important to note that often the constraints mentioned are calculated sequentially and in a two-dimensional shape, on a pair of decoupled dimensions, which limits the ability to optimize. For all this, a smoothing algorithm has been developed for a path planner that considers the complete three-dimensional kinematic constraints without decoupling the dimensions.

# Resum

Debut a la creixent popularitat sobre la varietat dels Vehicles No Tripulats tant en el camp militar com en el comercial, i de les seves capacitats per navegar per diversos entorns, ja siguin terrestres, aeris o marins, s'evidencia que la clàssica planificació de trajectòries i moviments bidimensionals 2D podria no ser suficient en un futur immediat. D'aquesta manera, s'ha de ressaltar que el present tesi aborda el problema dels Vehicles Aeris No Tripulats (UAV) d'ala fixa. En aquest sentit, la necessitat de trobar una trajectòria navegable en l'espai euclidià 3D es fa cada vegada més necessari. En el cas dels UAV, considerar la seva cinemàtica per generar trajectòries suaus en tres dimensions pot tenir un interès significatiu per a la navegació autònoma aèria. Finalment, els beneficis addicionals que es poden produir són importants.

La principal dificultat d'aquest problema és que els vehicles aeris de característiques no-holonómicas es veuen obligats a avançar sense la possibilitat de detenir-se a través de trajectòries 3D amb curvatures limitades. En aquest sentit, s'ha investigat la manera de proporcionar una completa caracterització de trajectòries òptimes per UAVs amb un radi de gir limitat que es mou en el pla tridimensional a una velocitat constant.

Per completar aquestes tasques, un planificador de trajectòries no només ha de proporcionar rutes tridimensionals per assolir una posició de destinació sense col·lisionar amb obstacles, sinó també ha d'assegurar que tal trajectòria sigui adequada per als UAVs que posseeixen propietats cinemàtiques específiques. Per tant, el desenvolupament de la feina ha completat la algorísmia que genera una trajectòria discreta tridimensional a l'definir un conjunt de punts 3D,

resultants d'una divisió de l'espai euclidià tridimensional de manera dinàmica, determinant les millors opcions d'avanç, evitant analitzar cada espai de l' entorn complet. D'aquesta manera, partint dels punts 3D resultants de la planificació de trajectòria tridimensional, s'ha generat una trajectòria en forma de corba suau construïda en funció de les limitacions de gir de l'UAV (ressaltant que és difícil assegurar que el camí resultant compleixi amb les restriccions cinemàtiques en les tres dimensions simultàniament). Finalment, és important destacar que sovint les restriccions esmentades es calculen seqöencialment i de forma bidimensional, sobre un parell de dimensions desacoblades, el que limita la capacitat d'optimització. Per tot això, s'ha desenvolupat un algoritme de suavitzat per a un planificador de trajectòries que considera les restriccions cinemàtiques tridimensionals completes sense desacoblar les dimensions.

# Índice general

# Capítulo 1

# Introducción

***Resumen:*** *Este capítulo introduce las principales ideas de la tesis, desarrollada en el marco de los vehículos autónomos AV, en concreto el trabajo ha sido desarrollado sobre Vehículos Aéreos No Tripulados UAVs. Es importante mencionar que el campo de los AVs ha experimentado un crecimiento constante en todo el mundo. Por lo tanto, con el fin de hacer un acercamiento hacia la motivación del tema de tesis, se ha realizado un breve resumen sobre la relevancia de los AVs y sus principales características. A continuación, se describe los objetivos planteados durante el transcurso del trabajo de tesis. Finalmente, el capítulo concluye haciendo una descripción de la estructura del documento de tesis.*

## 1.1 Motivación

El desarrollo de los *vehículos autónomos* (AV, del acrónimo en inglés *autonomous vehicles* ), por décadas ha tenido como objetivo el reemplazar a los pilotos humanos en diversas misiones. Las diferentes tecnologías de los AVs ya sean de clase terrestre (*Vehículos Terrestres Autónomos* "UGV", del acrónimo en inglés *Unmanned ground vehicle*), submarina (*Vehículos Submarinos Autónomos* "AUV", del acrónimo en inglés *Autonomous underwater vehicle*) o aérea (*Vehículos Aéreos Autónomos* "UAV", del acrónimo en inglés *Unmanned aerial vehicle*), presentan una constante evolución, por lo que su desarrollo tecnológico no se ha detenido, sin importar las soluciones que se presenten sobre

determinados problemas. Es así que, aun cuando han sido completadas soluciones a problemas específicos, tales soluciones han sido sometidas a nuevos cambios y modificaciones, lo cual evidencia que los problemas continúan en estado abierto y de ahí la necesidad de un desarrollo más profundo.

Los avances científicos de los AVs en los campos de detección [Maturana y Scherer 2015; Nevalainen y col. 2017], comunicaciones [B. Li y col. 2016; Amorim y col. 2017], sistemas de control [Yongqiang y col. 2009], etc; han sido por demás vertiginosos en los últimos años y han impulsado una constante explotación de la tecnología en la actualidad. Por otra parte, el desarrollo de tecnología AV implica una integración entre los diferentes tipos de vehículos y su correspondiente espacio de movimiento ya sea este espacio terrestre, marino o aéreo. Esta misma integración presenta una relevancia significativa que sigue presentando desafíos que deben ser resueltos y superados. Finalmente, es importante mencionar que el impacto económico que conlleva el desarrollo de AV, destacando exclusivamente la clase UAV en los Estados Unidos de América, supera los $13.6 mil millones de dólares, beneficio que se pronostica durará hasta el año 2025 y que, además, se prevé un incremento de 100.000 puestos de trabajo con un futuro impacto económico de $82 mil millones de dólares [Jenkins y Vasigh 2013].

Una misión típica de navegación sobre un AV, embarca los conceptos de diseño de trayectoria (planificación) y seguimiento de trayectoria (control). De esta forma, un AV debe poseer las capacidades para completar una trayectoria a través o muy cerca de un conjunto de waypoints[1] definidos dentro de un espacio de trabajo (workspace), al mismo tiempo que evita obstáculos siguiendo dicha trayectoria a lo largo del workspace. De hecho, debido a los estrictos requisitos operacionales y a las restricciones impuestas a los AVs por la autonomía, la seguridad, la eficiencia, etc; es difícil encontrar una solución completa para el guiado totalmente automatizado y el control de la navegación, por lo que la literatura, sugiere una división del problema en varios subproblemas, lo cual facilita la solución mediante una estructura de control jerárquica [McLain, Chandler y Pachter 2000; McLain y Beard 2005].

De esta manera, se puede hacer un acercamiento con referencia a la capacidad de navegación mencionada, destacando las habilidades de un piloto humano. En relación con esto, un piloto de aviación o piloto aviador debe poseer la capacidad de determinar y fijar una trayectoria que cumpla con el objetivo de alcanzar un destino definido (planificación global), en el supuesto de replanificación durante el trayecto, el piloto aviador ejecuta maniobras dinámicas

[1] Puntos de referencia tridimensionales utilizados en la navegación. Palabra que ha sido definida del inglés way (camino) y point (punto).

de corrección (planificación local). Por lo tanto, un planificador debe combinar la planificación local y global con el objetivo de completar una misión de navegación.

Así mismo, la navegación autónoma tiene como uno de sus objetivos el minimizar accidentes, ya sean producidos por los vehículos o por factores humanos y comúnmente esta tarea se aborda desde dos enfoques diferentes.

1) La completa sustitución del factor humano por máquinas en entornos de alto riesgo. Lo cual implica un funcionamiento casi por completo autónomo del vehículo, siempre existiendo la posibilidad de una manipulación remota. Siendo ejemplo de esto, el guiado automático utilizado en la industria.

2) La asistencia de la máquina en ciertas tareas. También conocido como control compartido o de asistencia, destacando que este campo mantiene un constante crecimiento en las últimas décadas. Siendo ejemplos relevantes los *Sistemas de Transporte Inteligente* (ITS, del acrónimo en inglés *Intelligent transportation system*) y los *Sistemas avanzados de asistencia al control* (ADAS, del acrónimo en inglés *Advanced driver-assistance systems*).

Una solución manual asistida mantiene ventajas sobre la navegación por completo manual y sobre una navegación por completo autónoma en situaciones de entornos difíciles (abarrotados de obstáculos). Una navegación asistida relaciona flexibilidad y entornos poblados de obstáculos, por lo que la algoritmia seguida en la tesis, no busca completar una navegación por completo autónoma, aplicando así una opción de enfoque con asistencia manual.

De esta forma, el trabajo de tesis se encamina en el estudio y desarrollo de planificación y navegación para UAVs en el workspace Euclidiano 3D. En la literatura existen grandes variedades de trabajos en los que se aborda el problema de planificación. No obstante, una base metodológica que prevalece en investigación realiza divisiones del entorno dimensional, mientras se buscan los espacios libres por los cuales el UAV pueda navegar. Este trabajo de tesis, se enfocará en la división del entorno como un espacio 3D discreto dinámico sobre el cual se determinarán las mejores opciones de ruta. Por otra parte, este primer proceso es insuficiente para la navegación de UAVs de ala fija (UAV de características no-holonómicas[2]) por lo que se ha necesitado un proceso poste-

[2]Es un sistema que se describe mediante un conjunto de parámetros sujetos a restricciones diferenciales.

rior de suavizado de curvas[3], de tal modo que las curvas sean aeronavegables. Del mismo modo, este último proceso ha sido planteado como un problema de optimización multiobjetivo.

## 1.2 Objetivos y estructura

El principal objetivo de la tesis ha consistido en el desarrollo de una nueva metodología de planificación de trayectoria para vehículos aéreos no tripulados en espacios de trabajo Euclidianos 3D. Al mismo tiempo, esta planificación contempla una planificación local y global con el objetivo de optimizar tiempo de cómputo, además de garantizar respuestas de trayectorias hacia el destino. A continuación, se describirán otros objetivos planteados durante el desarrollo del trabajo de tesis.

- Realizar un análisis y comparación de diferentes algoritmias en planificación de trayectorias 3D sobre UAVs, además de las técnicas de control utilizadas en navegación.
- Completar algoritmia de planificación de trayectorias y evasión de obstáculos.
- Implementar metodologías de planificación de trayectoria para UAVs, teniendo en cuenta las restricciones geométricas del entorno espacial, las dimensiones del UAV, además de las restricciones físicas implícitas del vuelo.
- Implementar metodologías de trayectorias suaves para el movimiento continuo de UAVs de ala fija, teniendo en cuenta limitaciones como la curvatura factible de vuelo y mínima distancia.
- Generar una trayectoria suave navegable por un UAV, al resolverlo como un *Problema de Optimización Multiobjetivo* (MOP, del acrónimo en inglés *Multi-objective optimization problem*) y agregando a este problema las restricciones cinemáticas y sin desacoplar dimensiones espaciales en pares.

De esta manera, la Figura 1.1, hace un resumen del trabajo realizado y de las etapas seguidas y culminadas. Primero, se completó un estudio de división del espacio de trabajo (workspace), del resultado de este primer estudio, en la segunda fase, se ha desarrollado un algoritmo de búsqueda de trayectorias libres

[3]Una curva suave es un mapa continuo "sin esquina" en el espacio n-dimensional, donde la palabra curva se interpreta en el contexto de geometría analítica.

de colisión por medio de una metodología determinista con base a una discretización del espacio euclídeo 3D. La tercera fase, toma los resultados anteriores y plantea la generación de una curva 3D que posea curvaturas y torsiones alcanzables por un UAV, al plantear un problema de optimización multiobjetivo MOP. Finalmente, la cuarta fase realiza un conjunto de simulaciones sobre los resultados de las etapas anteriores.

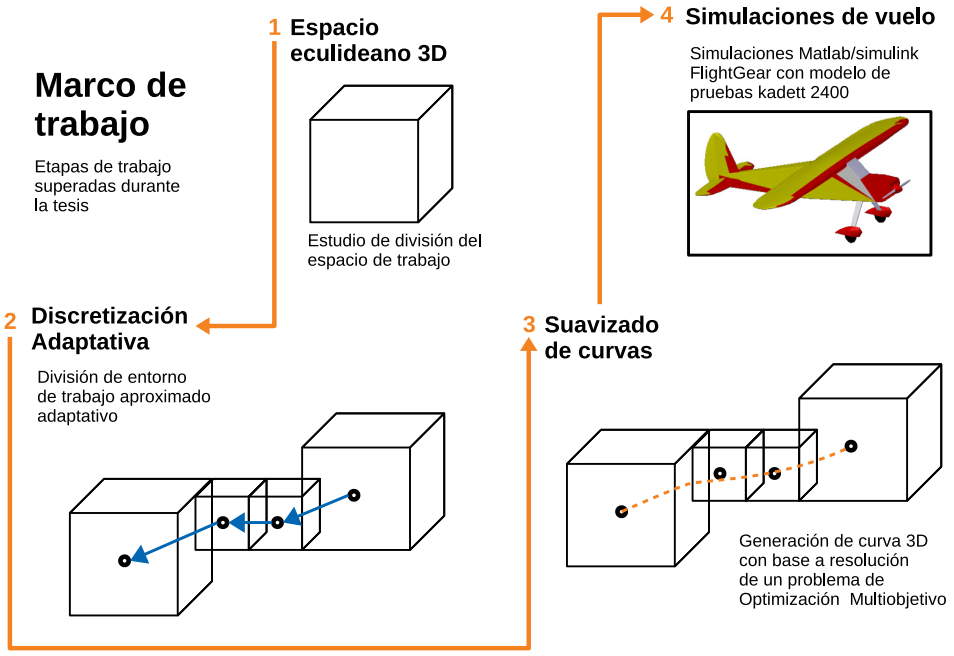



**Figura 1.1:** Fases de trabajo seguidas durante la tesis.

## 1.3 Estructura de la Tesis

Finalmente, se debe destacar que este documento de tesis contiene las transcripciones completas de cuatro publicaciones que exponen los resultados alcanzados como parte de los estudios de doctorado.

El material incluido en este documento, se encuentra organizado en dos partes. En la primera parte se presenta el estado del arte, el cual a su vez está dividido en cuatro capítulos:

- En el capítulo 2, se detallan varias definiciones sobre el concepto de planificación de trayectorias y planificación de movimiento, además de diferentes metodologías de división del entorno, así como varias perspectivas de solución a estos problemas. Finalmente, se hace mención sobre la relevancia en la problemática de UAVs de ala fija en planificación de trayectorias.
- El capítulo 3, estudia diferentes metodologías para la generación de trayectorias en forma de curvas continuas suaves, resolviendo el problema de navegación con diferentes clases de complejidad.
- El capítulo 4, analiza y estudia la metodología denominada optimización multi-objetivo MOP, su formalización matemática y la perspectiva usada para ser implementada en la problemática de generación de curvas continuas suaves.
- El capítulo 5, se realiza una breve recopilación teórica del modelo UAV *Kadett 2400* utilizado en el desarrollo de la tesis como modelo de simulación, se estudia su cinemática y dinámica además de sus limitaciones y alcances.

La segunda parte de esta tesis corresponde a las publicaciones resultantes de la misma, las cuales se dividen en cuatro diferentes documentos independientes, clasificados en artículos publicados en congresos y artículos publicados en revistas. Esta organización no busca imponer al lector sobre el conocimiento de un tema específico. Sin embargo, en esta sección se describirá las contribuciones de cada trabajo y como se relacionan entre ellos. Finalmente, se presentan las conclusiones finales y trabajos futuros relacionados al trabajo presentado.

- En el capítulo 6 se realiza un estudio de estado del arte sobre la planificación de trayectorias, haciendo énfasis en las metodologías de división del entorno y enfocando el esfuerzo sobre el estudio del tratamiento del espacio de trabajo (workspace) como un espacio tridimensional continuo y/o discreto.
- El capítulo 7 propone la creación de una trayectoria al realizar una división discreta del workspace cercano al UAV, se busca determinar espacios libres de colisión cercanos, conjunto de espacios que son manejados como un autómata finito discreto. Esta división del entorno determina la mejor posibilidad de desplazamiento del UAV al analizar un conjunto de recompensas sobre otro conjunto de posibles desplazamientos.
- En el capítulo 8, se divide el espacio de trabajo 3D en dos planos 2D, creando una curva continua 2D por cada plano, posteriormente se unen

estas dos curvas con el objetivo de construir una curva final continua 3D. Cada curva 2D es generada, tomando en consideración aspectos dinámicos (como la capacidad de maniobrabilidad) del UAV que son utilizados para las pruebas de simulación.

- En el capítulo 9, se propone la generación de una trayectoria de vuelo 3D, con el objetivo de minimizar el esfuerzo de giros verticales y horizontales de un UAV. La trayectoria se genera al unir un conjunto de segmentos lineales y esféricos, construyendo así una curva 3D continua sin separar los planos en pares. Es importante destacar que al generar segmentos lineales y/o esféricos, existirá infinitas de soluciones (en forma de curvas), por lo cual se propone una solución con base a una optimización multiobjetivo.
- Finalmente, en el capítulo 10, se mencionan las conclusiones finales y los posibles trabajos futuros sobre las temáticas abordadas a lo largo del trabajo de tesis presentado.

# Publicaciones

El trabajo desarrollado a lo largo de esta tesis ha generado las siguientes publicaciones:

*Artículos de Revista*

* Samaniego, F., Sanchis, J., García-Nieto, S., & Simarro, R. (2020). *Smooth 3D Path Planning by Means of Multiobjective Optimization for Fixed-Wing UAVs*. Electronics, 9(1), 51., Impact Factor: 2.412, JCR: Q2, SJR: Q1, DOI:https://doi:10.3390/electronics9010051.

* Samaniego, F., Sanchis, J., García-Nieto, S., & Simarro, R. (2019). *Recursive Rewarding Modified Adaptive Cell Decomposition (RR-MACD): A Dynamic Path Planning Algorithm for UAVs*. Electronics, 8(3), 306., Impact Factor: 2.412, JCR: Q2, SJR: Q1, DOI: https://doi.org/10.3390/electronics8030306.

*Artículos de Congreso*

* Samaniego, F., Sanchis, J., García-Nieto, S., & Simarro, R. (2018, October). *Comparative Study of 3-Dimensional Path Planning Methods Constrained by the Maneuverability of Unmanned Aerial Vehicles*. In 2018 7th International Conference on Systems and Control (ICSC) (pp. 13-20). IEEE.

* Samaniego, F., Sanchis, J., García-Nieto, S., & Simarro, R. (2017, October). *UAV Motion Planning and Obstacle Avoidance Based on Adaptive 3D Cell Decomposition: Continuous Space vs Discrete Space*. In 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM) (pp. 1-6). IEEE.

# Capítulo 2

# Planificación de trayectorias y de movimientos

***Resumen:*** *El presente capítulo hace una descripción sobre la formalización matemática y geométrica del problema de planificación de trayectorias, y se analizan diferentes metodologías estudiadas dentro del campo de la planificación de movimiento. Además, se hace un resumen sobre concepto de muestreo y sus diferentes clases. Por otra parte, se presentan las restricciones no-holonómicas en vehículos autónomos aéreos de ala fija.*

## 2.1 Introducción

En sus inicios, la robótica estuvo enfocada en el ámbito industrial. Sin embargo, se ha demostrado que la robótica ha sufrido un continuo desarrollo tecnológico, hasta tal punto que en la actualidad forma parte de la vida cotidiana del ser humano. Es evidente que un robot se diseña con el objetivo principal de facilitar las tareas cotidianas del ser humano en sus diferentes ámbitos como pueden ser domésticas [Jones 2006; Silvério y col. 2015], en rehabilitación [Díaz, Gil y Sánchez 2011], aprendizaje por imitación [Malekzadeh y col. 2014; Calinon 2018], inclusive la interacción social [Miguel A. Salichs y col. 2006] llegando a tal punto que un robot pueda incluso expresar emociones [Zecca y col. 2008;

Endo y col. 2008; Arkin, Ulam y Wagner 2011; Miguel A Salichs y Malfaz 2011].

En consecuencia, el desarrollo y evolución de esta rama de la tecnología, busca como uno de sus principales objetivos la autonomía[1]. Es así que, la autonomía se considera como una solución para una tarea 3D " *Dull, Difficult and Dangerous*" es decir: tareas tediosas, difíciles y peligrosas en el campo espacial. Por lo tanto, dentro de la robótica móvil de clase terrestre, acuática o aérea, se siguen presentando nuevos desafíos, relacionados con los materiales, las configuraciones ó el desarrollo de tareas de forma autónoma o independiente.

Existe un número importante de tareas que un robot autónomo puede completar, no obstante, nos centraremos en la rama de la robótica perteneciente a la aviación aérea no tripulada. Esta rama tecnológica ha sufrido un constante crecimiento en las últimas décadas, muestra de ello se menciona en [Cuerno-Rejado y col. 2016] donde se puede ver una evolución en sistemas de aviación, los cuales en sus inicios fueron diseñados con grandes dimensiones físicas y tecnología limitada. Sin embargo, con los años han ido evolucionando, tanto es así que en la actualidad han sufrido una importante disminución en sus características dimensionales y han aumentado de manera considerable su tecnología.

La planificación de trayectorias y la planificación de movimiento son tareas de relevancia significativa dentro del campo de la robótica móvil y de la automatización. La planificación de trayectorias genera una trayectoria geométrica desde un punto inicial hasta un punto final, pasando por un conjunto de puntos espaciales definidos dentro de un espacio de trabajo $\mathbb{W}$ (workspace[2]). Mientras la planificación de movimiento toma una trayectoria geométrica y considera diferentes propiedades cinemáticas y dinámicas de un vehículo autónomo, es decir fuerzas de inercia (pares), etc; de tal modo que las aceleraciones a lo largo de la trayectoria no alteren sus cinemáticas y dinámicas.

De esta manera, es importante destacar que diferentes enfoques han sido estudiados como [Latombe 2012; Choset y col. 2005; Kunchev y col. 2006; Steven M. LaValle 2006]. Finalmente, en las siguientes secciones se abordarán los conceptos mencionados, de modo que se llegue a una mejor comprensión sobre la temática.

[1] La autonomía, busca como objetivo la independencia del robot con respecto al control humano. Lo que significa que la necesidad de una constante presencia del ser humano es irrelevante, puesto que el sistema robótico debe poseer las habilidades de reacción frente a diferentes entornos estáticos o dinámicos.

[2] Las coordenadas del robot se especifican en el mismo sistema de coordenadas del mundo que está manipulando.

## 2.2 Planificación de trayectorias

**Definición 2.2.1** *Se entiende por* ***planificación de trayectorias*** *a la búsqueda de movimientos de un cuerpo, sin colisión desde un punto inicial* ($q_I$) *hasta un punto final* ($q_G$) *a través de un conjunto de obstáculos estáticos o móviles dentro un entorno conocido o dinámico.*

La planificación de trayectorias es una tarea perteneciente a la rama matemática de la geometría[3], pues se define como un camino geométrico en el que no se especifica ninguna ley de tiempo. Sobre la definición 2.2.1, la situación más sencilla implica obstáculos estáticos y un entorno conocido.

Dentro del campo de la robótica clásica, la planificación de trayectorias conlleva una carga computacional de tipo fuerte (NP-hard[4]).

Por otra parte, el concepto de planificación de trayectorias ha sido ampliamente usado en diferentes aplicaciones científicas adicionales a la navegación móvil autónoma, como por ejemplo en los campos de cirugía médica [Steffen y Zavattaro 2012], agricultura [Hameed 2014; Barrientos y col. 2011], operaciones militares y de rescate [Almurib, Nathan y T. N. Kumar 2011; Dubé y col. 2016], misiones espaciales [Tompkins, A. Stentz y Wettergreen 2004] o automatización industrial [Heping Chen, Fuhlbrigge y Xiongzi Li 2008; Pan y col. 2010].

Finalmente, se puede destacar que la algoritmia de planificación de trayectorias, lidia con autómatas que gobiernan diferentes números de articulaciones (manipuladores), diferentes restricciones (vehículos móviles) o incertidumbre (modelos imperfectos de entornos).

Esta sección describe un conjunto de conceptos necesarios para definir el problema geométrico de la planificación de trayectorias, los cuales serán detallados a continuación.

[3]Es una rama de las matemáticas que se ocupa del estudio de las propiedades de las figuras en el plano o el espacio.

[4]En teoría de la complejidad computacional, los problemas NP son problemas que se resuelven en tiempo polinómico, siendo NP el acrónimo en inglés de *nondeterministic polynomial time*.

### 2.2.1 Modelo matemático de la planificación de trayectorias

**Definición 2.2.2** *Un **estado** está definido por la mínima cantidad de información necesaria en determinado instante del tiempo para que, conociendo su entrada en ese mismo instante, se pueda determinar una variable del sistema en un instante posterior [Campoy y col. 2006].*

**Definición 2.2.3** *El **espacio de estados** es un espacio vectorial en el que el vector de estados toma valores, teniendo la misma dimensión que el número de elementos de dicho vector [Campoy y col. 2006].*

La teoría moderna de control se basa en la representación matemática de los sistemas de control por medio del concepto de estado. En ese sentido, una aproximación a un modelo de planificación de trayectoria matemático, puede ser definido utilizando un modelo en el espacio de estados [Steven M. LaValle 2006]. Entonces, dentro de un workspace $\mathbb{W}$, se puede presentar una situación denominada *estado* denotada por $x$, perteneciente al conjunto de situaciones o estados que conforman el *espacio de estados* definido por $X$ (en este trabajo se considera que el espacio de estados será finito). Por lo tanto, una acción $u$ puede ser aplicada a un estado actual $x$, lo cual producirá un nuevo estado $x'$ lo que genera una función de transición de estados $f$, siendo

$$x' = f(x, u) \tag{2.1}$$

Entonces, dado un estado $x$, $U(x)$ representa el conjunto de acciones aplicables a partir de $x$. En consecuencia, para distintos $(x, x' \in X)$, entonces $U(x)$ y $U(x')$, no son necesariamente disjuntos. Por lo tanto, el conjunto $U$ define las posibles acciones de los estados.

$$U = \bigcup_{x \in X} U(x) \tag{2.2}$$

donde, el conjunto de estados objetivo es definido por $X_G \subset X$, por consiguiente, el algoritmo de planificación de trayectoria debe encontrar la secuencia o conjunto finito de secuencias que determinen la transformación desde el estado inicial $x_I \in X_G$ hasta el estado final $x_G \in X_G$.

Esto produce un grafo de transición de estados[5], donde $X$ es representado por un conjunto de vértices y un conjunto de aristas que unen $x \in X$ hasta $x' \in X$, y serán agregados al grafo solo si existe una acción $u \in U(x)$ tal que $x' = f(x, u)$. Finalmente, es importante resaltar que los estados inicial $x_I$ y final $x_G$ son asignados como vértices en el grafo.

### *2.2.2 Planificación de trayectorias discreta*

Con el objetivo de definir un concepto de planificación que trayectorias matemática que optimice alguna clase de criterio (como distancia recorrida o consumo de energía), es necesario realizar un cálculo iterativo de funciones óptimas de costo-beneficio sobre el espacio de estados $X$.

**Definición 2.2.4** *Un **sistema de control en tiempo discreto** es un sistema en el que sus variables se pueden cambiar sólo en valores discretos de tiempo. La diferencia entre los sistemas de control en tiempo discreto y tiempo continuo consiste en que los sistemas en tiempo discreto están en la forma de datos muestreados (forma digital).*

La definición en sistemas discretos y continuos coincide, ya que el elemento $k_0$-*ésimo* corresponderá al instante $k_0 T$. Entonces, si se conoce el elemento $k_0$, dada la entrada $u(\lambda)$, con $k_0 \leq \lambda \leq k$, se puede determinar el estado del sistema.

**Definición 2.2.5** *La variable $k \in \mathbb{N}$ denota el tiempo discreto. Considérese los siguientes sistemas no lineales en tiempo discreto [Besselmann 2010]:*

$$x_{k+1} = f(x_k) \text{ (Sistema no controlado)} \tag{2.3}$$

$$x_{k+1} = f(x_k, u_k) \text{ (Sistema controlado)} \tag{2.4}$$

$$x_{k+1} = f(x_k, w_k), x_{k+1} = f(x_k, u_k, w_k) \left(\text{Sistema perturbado}\right) \tag{2.5}$$

donde $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, $w_k \in \mathbb{R}^{n_w}$ representan el estado del sistema, acciones de control y las incertidumbres, respectivamente. Las variables $u_k$,

[5]Un grafo es una estructura dinámica de datos que permite representar diferentes tipos de relaciones entre un conjunto de objetos llamados vértices (nodos) unidos por enlaces llamados aristas (arcos), que permiten representar relaciones binarias entre los elementos del conjunto de forma gráfica. Un grafo de transición de estados consiste en un conjunto de círculos que representan al conjunto de estados y un conjunto de segmentos de líneas dirigidas que representan el conjunto de transiciones entre los estados.

$w_k$ representan las entradas del sistema, donde $u_k$ será determinada por un controlador, mientras que $w_k$ es una entrada externa a la planta que en general, puede ser perturbaciones, ruido de proceso, incertidumbres en el modelo o el modo de operación del sistema.

En referencia al conjunto de ecuaciones pertenecientes a la definición 2.2.5, y haciendo una aproximación sobre la planificación de trayectorias, si se asume que la variable $k$ es expresada como un conjunto de pasos correspondiente a una secuencia $(u_1, u_2, \ldots, u_k)$ de acciones de $K$, y además, $k$, $x_I$ y $x_G$ han sido definidos, entonces la secuencia de estados $(x_1, x_2, \ldots, x_{k+1})$ pueden ser determinados usando una función de transición de estados. Asumiendo que $x_1 = x_I$ y que cada estado siguiente se obtiene a través de la ecuación 2.4, entonces se puede definir una función de costo por etapas aditivas como:

$$L(K) = \sum_{k=1}^{K} l(x_k, u_k) + l_F(x_F) \tag{2.6}$$

donde, la función de costo $l(x_k, u_k)$ produce un valor real por cada $x_k \in X$ y $u_k \in U(x_k)$. Del mismo modo, se define como $l_F(x_F) = 0$ siempre que $x_F = x_G$, en caso contrario $l_F(x_F) = \infty$. Por lo tanto, el objetivo es minimizar $L$, al construir una trayectoria en $K$ pasos, hasta alcanzar $l(x, u) \equiv 0$. Finalmente, con el fin de completar esta minimización, el proceso más común conlleva a generar todas las posibles secuencias de acciones de longitud $K$ y seleccionar la secuencia que produzca menor coste, lo cual requeriría un tiempo computacional elevado.

En este sentido, un planteamiento del problema con una base de grafos, puede ser resuelto eficientemente en tiempo polinómico. Sin embargo, la literatura describe una amplia variedad de problemas que pueden llevar a la generación de grafos de transición de estados muy grandes cuando se utilicen muy pocos predicados u operadores[6].

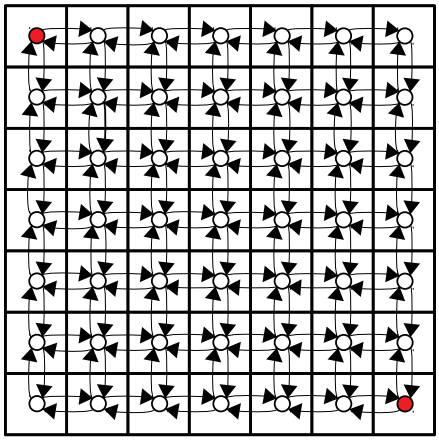
Un ejemplo relevante sobre la teoría descrita se detalla sobre el siguiente escenario. Sea un AV, el cual se mueve sobre un workspace bidimensional en forma de cuadricula (ver Figura 2.1), donde cada celda está representada por coordenadas $(i, j)$, el AV puede realizar movimientos discretos sobre la cuadrícula, siendo cada posible movimiento un único incremento o decremento en

[6]Un predicado es una función que puede conectarse con una o varias expresiones. Cuando un predicado se conecta a una expresión se dice que expresa una propiedad, en el caso que un predicado se conecta a dos o más expresiones se dice que expresa una relación. Es decir, los predicados son funciones que reciben un conjunto de argumentos, los procesan y devuelven como resultado diferentes valores.

las direcciones: arriba, abajo, izquierda o derecha, el objetivo es determinar la secuencia de acciones o el conjunto de secuencias de acciones que transforme desde el estado $x_I = (1, 1)$ hasta el estado final $x_G = (7, 7)$.

| | | | | | | |
|---|---|---|---|---|---|---|
| $(1,1)$ | $(1,2)$ | $(1,3)$ | $(1,4)$ | $(1,5)$ | $(1,6)$ | $(1,7)$ |
| $(2,1)$ | $(2,2)$ | $(2,3)$ | $(2,4)$ | $(2,5)$ | $(2,6)$ | $(2,7)$ |
| $(3,1)$ | $(3,2)$ | $(3,3)$ | $(3,4)$ | $(3,5)$ | $(3,6)$ | $(3,7)$ |
| $(4,1)$ | $(4,2)$ | $(4,3)$ | $(4,4)$ | $(4,5)$ | $(4,6)$ | $(4,7)$ |
| $(5,1)$ | $(5,2)$ | $(5,3)$ | $(5,4)$ | $(5,5)$ | $(5,6)$ | $(5,7)$ |
| $(6,1)$ | $(6,2)$ | $(6,3)$ | $(6,4)$ | $(6,5)$ | $(6,6)$ | $(6,7)$ |
| $(7,1)$ | $(7,2)$ | $(7,3)$ | $(7,4)$ | $(7,5)$ | $(7,6)$ | $(7,7)$ |

(**a**)Espacio de trabajo (workspace) en forma de cuadrícula.



(**b**)Grafo de transición de estados para un entorno en forma de cuadrícula.

**Figura 2.1:** Ejemplo de formalización del problema de planificación de trayectoria.

Por lo tanto, se puede definir un conjunto de acciones que representen los movimientos descritos posibles, siendo $U = (0, 1), (0, -1), (1, 0), (-1, 0)$ el conjunto de acciones (definido por cada par ordenado de coordenadas $(i, j)$) aplicables sobre cada estado $x$. En consecuencia, $U(x) = U \; \forall \; x \in X$. Finalmente, puesto que existe una relación directa entre los vectores estado-acción, una operación de suma completa la ecuación de transición de estados, definida como $f(x_k, u_k) = x_k + u_k$.

Una vez definida la ecuación de transición de estados, y asumiendo una distancia igual entre celdas contiguas verticales u horizontales, de acuerdo a la Figura 2.1(b), se puede apreciar que existe un conjunto de secuencias de acciones de longitud $K$ que transforme $x_I$ hasta $x_G$, de acuerdo a la ecuación 2.6 sobre el workspace $\mathbb{W}$ establecido. Finalmente, es importante destacar que la situación del entorno es ideal, al encontrarse libre de obstáculos.

Como es de esperar, este tipo de proceso matemático requiere un tiempo elevado hasta llegar a una solución, por lo que sería importante hacer un recorrido

en el espacio de estados, evitando realizar la completa exploración del conjunto finito de estados.

## 2.3 Planificación de movimiento

**Definición 2.3.1** *En teoría de control, la* ***planificación del movimiento*** *se refiere a la construcción de un conjunto de entradas para un sistema dinámico no lineal que pasa desde un estado inicial a un estado objetivo específico. La planificación de movimiento, se refiere al conjunto de movimientos de un robot en un workspace 2D o 3D que contiene obstáculos. Sin importar el modelo del robot o su configuración, el objetivo principal busca que un robot complete el conjunto de movimientos apropiados a su configuración de tal modo que evite la colisión hasta alcanzar el objetivo [Steven M. LaValle 2006].*

**Definición 2.3.2** *El* ***espacio de configuración*** *C-space o $C$, se considera como un espacio de estados especial, siendo el espacio de todas las posibles posiciones instantáneas de un sistema mecánico (robot), posee una estructura de variedad diferenciable de dimensión $N$, donde $N$ es el número de grados de libertad (DoF[7], del acrónimo en inglés Degrees of Freedom) del sistema. Además de movimientos de robots y variedades matemáticas, una definición apropiada del C-space proporciona una poderosa abstracción de modelos complejos y transformaciones [Lozano-Pérez y Wesley 1979].*

En la planificación de movimiento se realiza una transición desde, el espacio de estados hasta el C-space, estando la dimensión del espacio de configuración definida por el número de DoF del AV. Por lo tanto, la planificación de movimiento se traduce como una búsqueda de un camino continuo dentro del C-space.

La algoritmia basada en muestreo, ha demostrado ser una opción de resolución constante y efectiva sobre un amplio espectro de problemas difíciles de planificación, aunque sus garantías de integridad son débiles debido a que el espacio físico de búsqueda es muy grande, lo que implica hacer un recorrido costoso en tiempo. Sin embargo, la facilidad de aplicación de estas metodologías las hacen idóneas para la resolución del problema de planificación de movimiento.

[7] En ingeniería se refiere al número mínimo de parámetros necesarios a especificar para determinar completamente la velocidad de un mecanismo o el número de reacciones de una estructura.

### 2.3.1 Configuración geométrica del espacio

**Definición 2.3.3** *Un grupo ortogonal especial de transformaciones de preservación de distancia en el espacio euclídeo de dimensión N que contiene el elemento identidad se denota como* ***SO(N)****, y que consiste en todas las matrices ortogonales de determinante unidad. Por lo tanto, $SO(N)$ representa las matrices de rotación que definen la orientación de un cuerpo en relación a las coordenadas de la tierra.*

Una descripción geométrica tanto de un AV denotado como $A$, como del workspace $\mathbb{W}$ (el cual puede poseer dos tipos de entidades internas: AVs y obstáculos) debe ser especificada antes de completar la tarea de planificación de movimiento.

Si se asume que, $A$ se mueve en el espacio euclídeo ordinario $\mathbb{W} = \mathbb{R}^N$ (con $N = \{2, 3\}$), siendo $\mathbb{W} = \mathbb{R}^2$, entonces se puede definir un vector $q = (x, y)$, mientras que con $\mathbb{W} = \mathbb{R}^3$ se puede definir un vector $q = (x, y, z)$ (El símbolo $q$ se representa de forma estándar en la formulación Hamiltoniana de la mecánica clásica y en mecánica Lagraniana [Vershik y Faddeev 1995]). Donde $q$ representa un punto de configuración en el C-space, mientras $\dot{q} = dq/dt$ se refiere a velocidades. En consecuencia, C-space representa el conjunto de todas las posibles configuraciones de $q$ ($q \in C$).

Entonces, se puede definir una región de obstáculos $O$ dentro $\mathbb{W}$, como un conjunto cerrado $O \subset \mathbb{W}$, donde $O$ se expresa como una serie de figuras geométricas 2D o 3D. Por lo tanto, $A(q) \subset \mathbb{W}$ representa el conjunto de puntos espaciales ocupados por $A$, en todo el espacio de configuración $q \in C$.

De esta manera, la región de obstáculos dentro de $C$ está definido por el conjunto cerrado $C_{obs} = \{q \in C | A(q) \cap O \neq 0\}$. Mientras que la región libre de colisión es el subconjunto abierto dentro de $C$ tal que $C_{free} = C \setminus C_{obs}$. Por tanto, la configuración ($q \in C$) es el espacio de todas las configuraciones, siendo:

- Si $A$ se representa como un punto en el espacio, entonces este se desplaza en $\mathbb{W} = \mathbb{R}^2$, por tanto, C-space será un plano y se puede representar una configuración espacial que utiliza dos parámetros $(x, y)$.
- Si $A$ tiene una forma 2D, entonces $A$ se puede trasladar y rotar, por tanto $\mathbb{W} = \mathbb{R}^2$. Sin embargo, el C-space pertenece a un grupo especial euclídeo $SE(2) = \mathbb{R}^2 \times SO(2)$ donde $SO(2)$ es un grupo ortogonal especial de rotaciones 2D, y se puede representar una configuración espacial usando 3 parámetros $(x, y, \theta)$.

- Si $A$ tiene una forma 3D sólida, entonces $A$ se puede trasladar y rotar, en consecuencia $\mathbb{W} = \mathbb{R}^3$, sin embargo, el C-space pertenece a un grupo especial euclideo $SE(3) = \mathbb{R}^3 \times SO(3)$, por consiguiente esta configuración requiere de 6 parámetros: siendo 3 para la traslación $(x, y, z)$, y 3 para las rotaciones $(\alpha, \beta, \sigma)$ (ángulos de Euler).

El problema espacial puede incrementar en dimensiones $N > 3$, si se toma en cuenta el espacio de estados o configuraciones adicionales de $A$ (como por ejemplo las restricciones internas de movimiento).

### *2.3.2 Muestreo del espacio basado en planificación de movimiento*

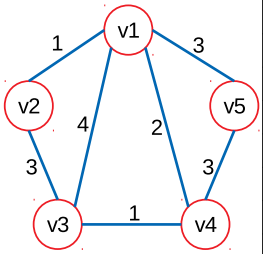
Una tarea trascendente en planificación de movimiento es la determinación de espacios navegables y no navegables (también definidos como *muestreo espacial*) dentro de $\mathbb{W}$.

El muestreo espacial consiste en escoger un conjunto finito de muestras representativas en $\mathbb{W} = \mathbb{R}^N$, este enfoque ha demostrado ser exitoso en la práctica para problemas complejos. Una metodología de muestreo conlleva la generación de una gran secuencia de muestras, lo cual implica que el límite de muestras tiende a infinito. Por otra parte, cada nueva muestra generada se aproxima arbitrariamente al punto objetivo. En este sentido, siempre se puede construir un conjunto de muestras único a partir de otra secuencia de muestras, no obstante, se pueden construir muchas secuencias alternativas a partir del mismo conjunto de muestras.

De este modo, el muestreo se basa en la reducción del C-space N-dimensional a un conjunto de caminos unidimensional, típicamente representado como un grafo computacional [Archdeacon 1996] $G = (V, E)$. Donde $G$ es una pareja ordenada en la cual $V$ representa un conjunto de vértices y $E$ representa un conjunto de aristas. Siendo, $V$ un conjunto no ordenado de coordenadas que indican las ubicaciones espaciales libres de colisión $C_{free}$ tal que $\{x, y\} \in E : \mathbb{R}^2$ ó $\{x, y, z\} \in E : \mathbb{R}^3$. Mientras que el conjunto $E$ representa las conexiones visibles entre otro conjunto $V$. Por lo tanto, $G$ se representa por medio de líneas no orientadas que unen $V$ (ver Figura 2.2). En este sentido, se debe destacar que las metodologías de muestreo requieren de una función métrica de distancia [Cha 2007] sobre el C-space, con el objetivo de determinar el conjunto $V$ que conecte las nuevas configuraciones dentro del vecindario[8]. Finalmente, hay que recalcar que el objetivo de las metodologías de muestreo se enfoca en

[8]El número de enlaces a un mismo vértice, enlaces definidos en base a una función (generalmente una métrica de distancia).

generar un grafo $G$ reducido que llegue a una gran cantidad de puntos posibles $C_{free}$ dentro de $\mathbb{W}$.




| *Conjuntos de aristas* | *Pesos* |
|---|---|
| v1 - v2 | 1 |
| v1 - v3 | 4 |
| v1 - v4 | 2 |
| v1 - v5 | 3 |
| v2 - v3 | 3 |
| v3 - v4 | 1 |
| v4 - v5 | 3 |

(**a**)Grafo ponderado con 5 vértices (círcunferencias rojas) y 7 aristas (líneas azules). En el grafo se puede ver etiquetas que indican la unión entre pares de vértices y las ponderaciones indicadas en las aristas que unen cada par de vértices.

Matriz de adyacencia

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(**b**)Matriz cuadrada $M$ de tamaño $n^2$, donde $n$ es el número de vértices. En el caso de existir una arista entre dos vértices, entonces el elemento $m_{x,y}$ es 1, en caso contrario, es 0.

**Figura 2.2:** Ejemplo de grafo computacional.

Por otra parte, una función métrica de distancia euclídea en $\mathbb{R}^n$ aplicable sobre el C-space y que permite convertir un espacio topológico $X$ en un espacio métrico, puede ser definida como $(X, \rho)$. Siendo $X$ al que se aplica una función $\rho : X \times X \to \mathbb{R}$. Finalmente, la métrica más relevante sobre $\mathbb{R}^n$ para esta clase de problemas de búsqueda está dada por [Ball 1990; Karlovitz 1970]:

$$\rho(x, x') = \left( \sum_{i=1}^{n} |x_i - x'_i|^p \right)^{1/p} \tag{2.7}$$

donde la función $\rho$ define la distancia entre puntos en una métrica espacial para cualquier valor de $p \geq 1$.

Debido a que el trabajo realizado durante esta tesis se basa en muestreo de espacio, se han dividido las principales estrategias de muestreo espacial en dos grupos, siendo estos aleatorio (espacio continuo) y determinista (espacio discreto), metodologías que serán descritas a continuación.

### 2.3.3 Muestreo aleatorio

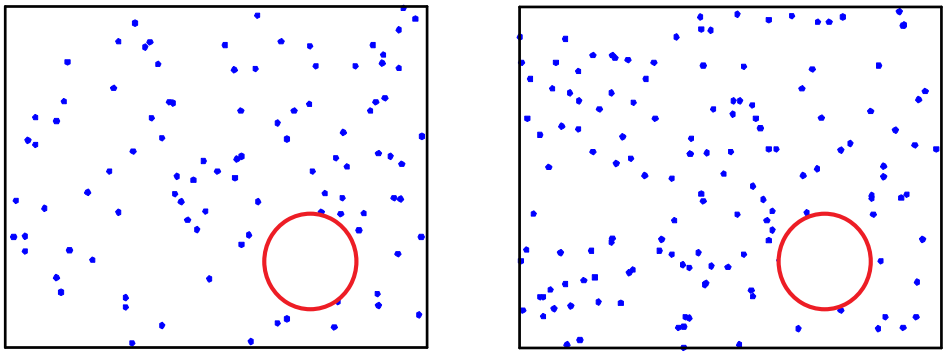
Un muestreo aleatorio uniforme lleva como objetivo, alcanzar una densidad de probabilidad uniforme (probabilidad uno), lo cual asegura la integridad de probabilidad sobre el algoritmo de planificación. Un espacio $C_{free}$ se construye a partir de productos cartesianos, siendo las muestras aleatorias independientes [Steven M. LaValle 2006]. De esta manera, si $X = X_1 \times X_2$, entonces sus correspondientes muestras aleatorias uniformes pueden ser definidas como $x_1$ y $x_2$, mismas que serán tomadas de $X_1$ y $X_2$ respectivamente. En consecuencia, la coordenada espacial aleatoria $(x_1, x_2)$ representa la muestra aleatoria uniforme sobre $X$. Es decir, las muestras se seleccionan de manera aleatoria sobre el C-space.

De esta manera, el objetivo es construir un conjunto de puntos cartesianos generados por la aleatoriedad y unirlos por medio de una métrica, la misma que indicará la vecindad de cada punto generado desde $q_I$ hasta $q_G$. En consecuencia, el conjunto de puntos espaciales $C_{free}$ son agregados a un grafo de conectividad $G$ con el fin de determinar una trayectoria entre $q_I$ y $q_G$. Finalmente, se debe destacar que la característica de aleatoriedad uniforme consigue cubrir gran parte del C-space.

Sin embargo, el muestreo aleatorio presenta un problema importante, debido a la propia naturaleza de la aleatoriedad, sobre un $\mathbb{W}$ se puede obtener un elevado número de muestras en ciertas subregiones, mientras que en otras del mismo $\mathbb{W}$ el número de muestras puede ser bajo.

La irregularidad mencionada, se puede ver en la Figura 2.3, el ejemplo es ilustrado en un mapa de 2 dimensiones, en el que se han generado 2 conjuntos de muestras aleatorias como puntos de color azul. Del mismo modo, se puede distinguir una circunferencia en rojo dentro de $\mathbb{W}$ que denota un obstáculo.

Por otra parte, es importante resaltar que incluso las muestras aleatorias presentan cierto determinismo, ya que obedecen a un algoritmo matemático que responde a pruebas estadísticas como por ejemplo chi-cuadrado [Miller y Siegmund 1982] que mide de manera estadística la distancia del valor calculado esperado.



(**a**)100 Muestras aleatorias. (**b**)150 Muestras aleatorias.

**Figura 2.3:** Muestreo aleatorio e irregularidad en el C-space, en 2 grupos de muestras aleatorias sobre $\mathbb{W} = \mathbb{R}^2$ con un obstáculo. Las muestras son generadas dentro de $\mathbb{W}$ pero fuera de la zona del obstáculo.

En muchos casos, obtener una solución al problema de planificación de trayectoria al aplicar muestreo aleatorio se completa de manera sencilla, al aplicar una medida Haar [Hahn 1978]. Sin embargo, al presentarse un $SO(3)$ complejo, se utiliza un cuaternio aleatorio con respecto a Haar, siendo un ejemplo de esto

$$h = (\sqrt{1-u_1}\sin(2\pi u_2), \sqrt{1-u_1}\cos(2\pi u_2), \sqrt{u_1}\sin(2\pi u_3), \sqrt{u_1}\cos(2\pi u_3)) \tag{2.8}$$

donde $(u_1, u_2, u_3 \in [0,1])$ son muestras aleatorias uniformes dentro de $\mathbb{W}$. Sin embargo, se necesita de una cierta medida de irregularidad para obtener una solución satisfactoria, como se menciona en [Beck y col. 2012].

A continuación, se describirán un conjunto de metodologías de planificación basadas en muestreo aleatorio.

**Planificador aleatorio de trayectorias (RPP, *Randomized Path Planning* [Caselli y Reggiani 2000; Caselli, Reggiani y Rocchi 2001]):** Diseñado para resolver problemas de planificación de trayectorias en espacios de configuración de grandes dimensiones. El planificador genera una resolución probabilista completa en un tiempo ilimitado, lo que garantiza encontrar una solución "si existe". Sin embargo, en la práctica un planificador debe finalizar su proceso en un tiempo limitado.

**Mapas probabilísticos (PRM, *Probabilistic roadmap*** [Denny y Nancy M Amatoo 2013a; Q. Li y col. 2014a; Bhattacharya y Gavrilova 2008]**):** Este método busca crear un mapa, partiendo de la generación de puntos aleatorios $C_{free}$. Este mapa conecta $A$ con los puntos objetivo.

**Arboles de exploración rápida (RRT, *Rapidly-exploring random tree*** [Evestedt y col. 2015; Navarro-Perez y Buitrago 2016; Nazif, Iranmanesh y Mohades 2008; Jaillet, J. Cortés y Siméon 2008]**):** Este método tiene como objetivo crear árboles que se expanden a partir de la generación de puntos aleatorios los cuales conectan los puntos objetivo.

**Métodos basados en optimización estocástica (*Stochastic optimization*)** [Kurt Marti y S. Qu 1998; Marti 1999]: Este conjunto de métodos, por lo general, utilizan diferentes clases de análisis bio-inspirados con el objetivo de llegar a una optimización estocástica. Cabe resaltar que este tipo de enfoque es útil en espacios de búsqueda muy amplios.

**Algoritmos genéticos (GA, *Genetic algorithm*** [Ismail, Sheta y Al-Weshah 2008; Tuncer y Yildirim 2012; Tsai, H.-C. Huang y Chan 2011]**):** Inspirados en la evolución genética. Parten del principio de evolución de una población de individuos sometidos a acciones aleatorias (mutaciones y recombinaciones genéticas) y selección de individuos aptos de supervivencia y descartando los menos aptos.

**Colonia de hormigas (ACO, *Ant colony optimization algorithms*** [Duan, Yu y col. 2010; Xiong Chen y col. 2013; Châari y col. 2012; Cekmez, Ozsiginan y Sahingoz 2014]**):** Esta metodología está inspirada en el comportamiento colectivo de las hormigas en su búsqueda de alimentos. Esta metodología se resume en la determinación probabilística sobre un problema computacional al determinar los mejores caminos dentro de grafos.

**Enjambre de partículas (PSO, *Particle swarm optimization*** [Masehian y Sedighizadeh 2010; H.-C. Huang y Tsai 2011; Raja y Pugazhenthi 2009; Roberge, Tarbouchi y Labonté 2012]**):** Esta metodología es utilizada en espacios de búsqueda con grandes dimensiones, está inspirada en el comportamiento de enjambres de insectos en la naturaleza. La metodología, sitúa partículas al azar en el espacio de búsqueda, pero dándoles la posibilidad de que se muevan a través de él, de acuerdo a un conjunto de reglas que tienen en cuenta el conocimiento personal de cada partícula y el conocimiento global del enjambre.

Una vez realizada la descripción básica sobre el funcionamiento del muestro aleatorio, y además de mencionar varias metodologías con este fundamento teórico, en la sección 6.3 se hace un resumen adicional sobre 2 metodologías importantes de la literatura en planificación de trayectorias, siendo la primera, Mapas de ruta probabilistas (PRM), descrita en la sección 6.3.1, del mismo modo se hace una breve descripción de la Exploración rápida de árboles (RRT) en la sección 6.3.2. Es importante destacar que los resúmenes de las metodologías mencionadas han sido completados con un enfoque 3D.

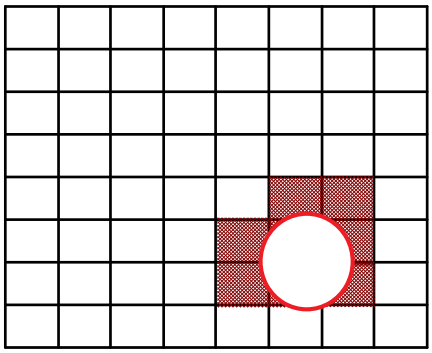
### *2.3.4 Muestreo determinista*

Un muestreo determinista busca una integridad en la solución del problema de planificación de trayectoria, lo cual significa que, si una solución a un problema existe, esta solución será encontrada, en el caso contrario el algoritmo podría permanecer en un estado de ejecución de manera infinita.

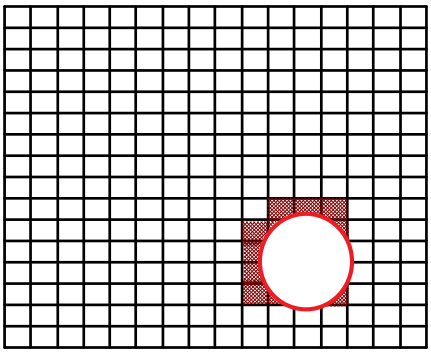
En términos generales, este conjunto de métodos consiste en construir un grafo que interconecte el punto de inicio $q_I$ con el punto objetivo $q_G$, como siguiente acción se determina la trayectoria más adecuada considerando algún índice de desempeño, por lo general la distancia más corta.

Un ejemplo de muestreo determinista se basa en construir una malla, generalmente en forma de cuadrícula, como se puede ver en la Figura 2.4. El objetivo es definir un conjunto de muestras $C_{free}$ (en forma de figuras geométricas cuadradas) de tal modo que se cubra la mayor parte posible del C-space. Al definir una malla con una resolución establecida, la resolución de la cuadrícula establece el tamaño de paso de cada eje. En consecuencia, al disminuir el tamaño de la cuadrícula, la resolución aumenta, lo cual implica una generalización de resolución completa.

Se puede apreciar en la Figura 2.4(a) que el número de celdas es $k = 64$, mientras que en la Figura 2.4(b) el número de celdas es $k = 256$. Se observa que el entorno se particiona en una malla de cuadros de igual tamaño, definiendo una dispersión dentro de $X = [0, 1]^n$. En consecuencia, el número de cuadros por eje se define por $[k^{1/n}]$, siendo $n$ un número real entero. Entonces, $[k^{1/n}]$ retorna el número de cuadros (muestras) por eje para una malla, siendo $k$ muestras en $n$ dimensiones (malla resultante denominada "*malla de Sukharev*" [Sukharev 1971]).



(**a**)Malla (8 $filas \times 8$ $columnas$), 64 celdas.



(**b**)Malla (16 $filas \times 16$ $columnas$), 256 celdas.

**Figura 2.4:** Muestreo determinista en forma de malla. Dentro de dicho entorno en las Figuras 2.4(a) y 2.4(b), se puede apreciar un obstáculo en forma de circunferencia de color rojo. Los cuadros que interceptan con el obstáculo son celdas con colisión (marcados como una subcuadrícula diagonal en rojo) y por tanto son no transitables, lo que implica, descarte de celdas en la búsqueda de posibles trayectorias.

Por lo tanto, el límite inferior para cualquier conjunto $P$ de $k$ muestras se define como:

$$\delta(P) \geq \frac{1}{2\lfloor k^{1/d} \rfloor} \tag{2.9}$$

Lo que significa que mantener una dispersión fija, implica un crecimiento exponencial de muestras en función de la dimensión $d$.

Esta metodología de tipo malla permite una distribución uniforme de los puntos dentro del C-space, ubicando así cada punto $C_{free}$ en el centro de cada celda, lo cual es una buena opción para resolver el problema de planificación de trayectoria. En este sentido, se busca definir una metodología que construya una estructura finita de datos que codifique el problema de planificación de trayectoria.

Dentro del muestreo determinista, esta algoritmia se conoce como *descomposición de celdas*. Entonces, una descomposición de celdas divide el C-space con el objetivo de determinar el conjunto finito $C_{free}$. Por lo tanto, este conjunto $C_{free}$ debe contener los puntos $q_I$ y $q_G$. Finalmente, el conjunto $C_{free} \in$ C-

space permite construir una trayectoria o un conjunto de trayectorias desde $q_I$ hasta $q_G$.

Una metodología de descomposición de celdas, determina el espacio libre dentro del C-space por el cual $A$ puede navegar, divide las regiones en celdas $C_{free}$ y $C_{obs}$. Lo cual permite la construcción de una estructura de tipo grafo $G$, que contiene información relevante del conjunto de celdas $C_{free}$ y $C_{obs}$ (tamaño de celdas, ubicación, además de celdas vecinas[9]). Finalmente, los vértices del grafo $G$ representan el conjunto de celdas $C_{free}$ y la conexión entre $C_{free}$ que comparten límites (adyacencia entre $C_{free}$), se representa por medio de aristas.

A continuación, se realiza una breve descripción de metodologías de planificación con base a un muestreo discreto.

**Grafos de visibilidad (*Visibility graph* [H.-P. Huang y Chung 2004; Kaluđer, Brezak y Petrović 2011; Ma, G. Zheng y Perruquetti 2013; Majeed y Lee 2018]):** Proporcionan un enfoque geométrico para resolver el problema de planificación de trayectorias. Este método requiere de un entorno definido con polígonos, tanto en el plano como en el espacio 3D. La representación del grafo se determina aplicando el concepto de visibilidad, el cual consiste en determinar si dos puntos sobre el espacio se pueden unir mediante un segmento rectilíneo que no presente interferencia con ningún obstáculo.

**Diagramas de Voronoi (*Voronoi diagram* [Garrido y col. 2006; Bhattacharya y Gavrilova 2007; Tong y col. 2012]):** Es una estructura computacional que almacena toda la información referente a la proximidad entre puntos. Dado un conjunto de puntos $\{p_1, \ldots, p_n\}$, un diagrama Voronoi realiza una división del plano en subregiones, siendo la región $i$ el conjunto de puntos más cercanos a los $p_i$ que a cualquier $p_j, j \neq i$.

**Modelado del espacio libre (*Free space modeling* [Meng 1988; Buniyamin y col. 2011]):** Esta metodología se aplica a espacios de trabajo con obstáculos poligonales. La construcción del grafo se realiza mediante cilindros rectilíneos generalizados, con el objetivo de determinar una trayectoria para $A$, de tal manera que se desplace lejos de los obstáculos.

**Planificación de trayectorias empleando campos potenciales (*Potential fields path planning* [Vadakkepat, Tan y Ming-Liang 2000; Mora y Tornero 2008]):** El fenómeno de los campos potenciales basa su

[9] Vecindario es la representación de las celdas de espacio libre $C_{free}$ que comparten límites dentro del workspace.

funcionamiento en el cálculo de campos imaginarios de repulsión que emanan los obstáculos. Estos campos varían de acuerdo a la distancia del obstáculo y pueden ser representados de forma geométrica, además de la posibilidad de establecer límites de influencia de los obstáculos para eliminar cálculos innecesarios sobre obstáculos distantes.

El método de campos potenciales resulta ser una de las técnicas más populares en la generación de trayectorias para robots móviles por ser una forma eficiente de solucionar este problema.

Con la descripción del funcionamiento básico de muestreo discreto, además del extracto teórico presentado de varias metodologías con este fundamento, en la sección 6.4 se hace un breve resumen sobre dos metodologías en planificación de trayectoria, siendo la primera: Mapas de ruta probabilistas con descomposición de celdas exactas (PRM-ECD), descrita en la sección 6.4.1. La segunda hace una breve descripción sobre una descomposición aproximada de celdas modificada en la sección 6.4.2. Es importante destacar que los resúmenes de las metodologías mencionadas han sido completados con un enfoque 3D.

## 2.4 Problemática de los UAVs en el espacio 3D

Antes de detallar la problemática de los UAVs de clase no-honolómico en el espacio 3D, se hará una breve descripción sobre la problemática holonómica. Del mismo modo se hará una breve descripción sobre las restricciones no-holonómicas definiendo ejemplos de vehículos móviles de este tipo.

Sin importar los avances tecnológicos en el campo de los AVs, uno de los principios básicos a solventar es el conocimiento propio sobre su localización, su destino, además del conocimiento de su entorno (odometría [Mohamed y col. 2019]). Con estos aspectos determinados de manera clara, se puede guiar al AV en un escenario mediante una planificación de trayectorias que supere obstáculos sin realizar colisiones [Steven M. LaValle 2006].

### 2.4.1 Restricciones holonómicas

Un robot es capaz de manejar 3 DoF (con respecto al espacio euclídeo de movimiento), que caracterizan los movimientos de desplazamiento en el eje $x$, desplazamiento en el eje $y$ y rotaciones entorno al eje $z$ por separado.

En este sentido, las restricciones holonómicas son todas las restricciones a las que está sometido un sistema robótico y que pueden ser integrables, es decir que son restricciones posicionales de la forma:

$$f(q_1, q_2, \ldots, q_n, t) = 0 \tag{2.10}$$

donde las variables $q_i$ son coordenadas del sistema y representan las limitaciones asociadas con el número de DoF controlables, siempre que no se encuentren sometidos a restricciones no-holonómicas[10]. Estas limitaciones pueden expresarse matemáticamente mediante ecuaciones que contienen sólo variables de posición.

En consecuencia, se puede definir a un AV de características holonómicas con relación a la movilidad, destacando su capacidad de modificar su dirección de forma instantánea (considerando una masa despreciable), sin necesidad de realizar una rotación previa. Por otra parte, cabe destacar que la definición mencionada no especifica el entorno de movimiento, involucrando así, a AVs terrestres, marinos o aéreos.

### 2.4.2 Restricciones no-holonómicas

Una restricción no-holonómica es aquella que no puede expresarse únicamente en términos de las variables de posición, sino que incluye la derivada temporal de una o más de esas variables. Estas ecuaciones de restricción no pueden integrarse para obtener relaciones únicamente entre las variables conjuntas. El ejemplo más común en los sistemas de vehículos terrestres surge del uso de una rueda o rodillo que rueda sin deslizarse sobre el suelo (en el caso de vehículos móviles de ruedas). Por otra parte, si se hace una referencia para los vehículos aéreos de ala fija, estas restricciones hacen mención a la capacidad de realizar maniobras de vuelo, pues sobre esta clase de vehículos aéreos no existe la posibilidad de detenerse y hacer un movimiento, sino mantener la velocidad y ejecutar una determinada maniobra sobre el espacio aéreo. Por otra parte,

[10] Son aquellas que impiden modificar la dirección instantánea del AV, por ejemplo, las ruedas de un AV terrestre.

un deslizamiento o vibración, puede ser traducido debido al efecto del viento o fenómenos atmosféricos sobre el AV aéreo.

### *Vehículos Aéreos de Ala Fija*

Un Avión (Vehículo Aéreo de Ala Fija o Aeroplano) es un tipo de aeronave cuya contextura lo hace más pesado que el aire, capaz de generar sustentación por sus propios medios. Sus características físicas más salientes resultan ser las alas que tiene dispuestas a sus costados y su propulsión es ejercida por uno o más motores.

Aunque los vehículos aéreos son diseñados con diferentes propósitos, la mayoría de ellos tienen componentes estructurales similares, aunque estos componentes pueden variar en función del uso específico para el cual son diseñados.

Las superficies de mando y control modifican la aerodinámica del vehículo aéreo (ver Figura 2.5) provocando un desequilibrio de fuerzas, donde una o más de ellas cambian de magnitud. Este desequilibrio, es lo que hace que el vehículo aéreo se mueva sobre uno o más de sus ejes, incremente la sustentación, o aumente la resistencia.

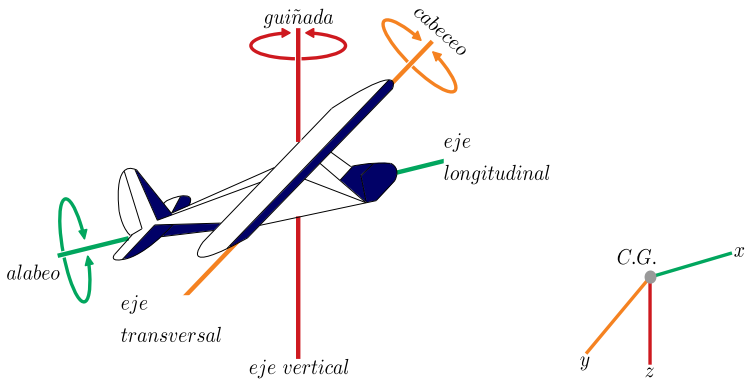



**Figura 2.5:** Los tres ejes de movimiento del avión (longitudinal, transversal y vertical) y sus rotaciones de alabeo, cabeceo y guiñada.

Se llama alabeo (del inglés *roll*) al movimiento sobre el eje longitudinal (eje de alabeo), cabeceo (del inglés *pitch*) al movimiento sobre el eje transversal (eje de cabeceo) y guiñada (del inglés *yaw*) al movimiento sobre el eje vertical (eje de guiñada). Los alerones producen movimientos de alabeo debido a la diferencia

de sustentación producida por el movimiento asimétrico. Los alerones están situados en la parte posterior del extremo del ala, y se accionan girando a un lado u otro el volante o palanca de mando. El timón de profundidad, situado en las superficies de estabilización horizontal de la cola, provoca el movimiento de cabeceo cuando se tira o se empuja del volante o palanca de mando. El movimiento de cabeceo del avión provoca la modificación del ángulo de ataque. El mando de control del timón de profundidad es el mando de control del ángulo de ataque. La guiñada es producida por el movimiento del timón de dirección, situado en las superficies de estabilización vertical de la cola, al accionarse los pedales de control.

Un vehículo aéreo ya sea en despegue, aterrizaje o en vuelo, debe solventar un conjunto de diferentes tareas o maniobras fundamentales, y están directamente relacionadas con:

1) La altura, (ascender, descender, o mantener una altitud constante).

2) La dirección (giros de derecha, izquierda, o mantener una dirección de vuelo).

3) La velocidad (acelerar, desacelerar o mantener una velocidad constante).

Conjunto de maniobras fundamentales que en vuelo se emplean individualmente o combinadas.

Dentro del espacio cartesiano 3D la posición del vehículo aéreo se denota como $p\{x, y, z\}$, mientras que el escalar de velocidad se denota como $v$. Las restricciones cinemáticas se modelan en base a las ecuaciones cinemáticas de movimiento como se menciona en [W. Cai, M. Zhang e Y. R. Zheng 2017; Paranjape y col. 2015; Hobby 1986].

En este sentido, un UAV presenta un sistema de coordenadas cuerpo fijas con seis DoF, por lo tanto, su movimiento puede ser descrito en relación a un marco de referencia inercial fijo. En la Figura 2.6 se muestra la ubicación y el movimiento de un UAV en el espacio cartesiano tridimensional, donde su posición se denota como $p\{x, y, z\}$, y su dirección de movimiento se denota como $\Phi = \{\phi, \theta\}$, donde $\theta$ y $\phi$ son el ángulo del plano $X - Y$ y el ángulo del eje $Z$ proyectados a partir de dicha dirección. El escalar de velocidad del UAV se denota como $F$, el ángulo del plano $X-Y$ proyectado y el ángulo del eje $Z$ están limitados, haciendo que su movimiento tenga restricciones no-holonómicas [W. Cai, M. Zhang e Y. R. Zheng 2017] y se expresan como:

$$\begin{aligned} \dot{x} &= F\cos\theta\cos\phi \\ \dot{y} &= F\cos\theta\cos\phi \\ \dot{z} &= F\sin\theta \end{aligned} \tag{2.11}$$

donde el operador de punto es la derivado, y

$$\begin{aligned} \dot{\phi} &= \zeta,\ \zeta \in [-\omega_1, \omega_1] \\ \dot{\theta} &= \Theta,\ \Theta \in [-\omega_2, \omega_2] \end{aligned} \tag{2.12}$$

donde $\omega_1$ y $\omega_2$ representan los límites de la curvatura. Las restricciones no-holonómicas requieren que las trayectorias del UAV satisfagan las continuidades $G^0$ y $G^1$ [K. Yang y Sukkarieh 2010; B. A. Barsky y T. D. DeRose 1990], continuidades que son detalladas en el capítulo 3.

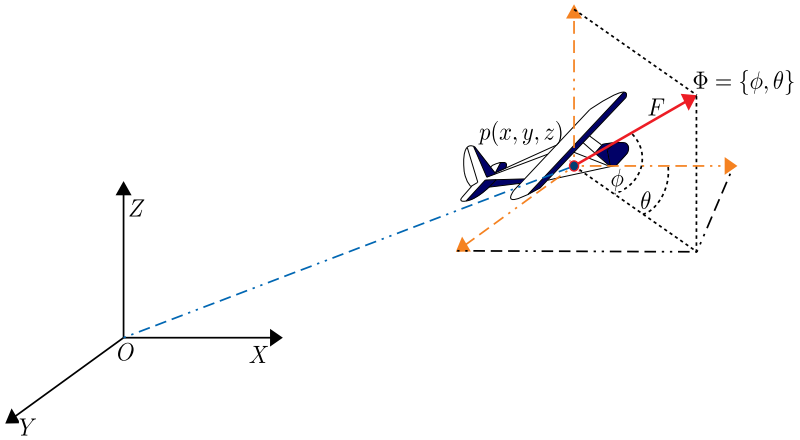



**Figura 2.6:** Dirección de movimiento del UAV en el espacio cartesiano euclídeo en 3D.

## 2.5 Conclusiones del capítulo

El objetivo de este capítulo fue proporcionar una revisión teórica, necesaria para permitir una representación matemática en varios aspectos como: la planificación de trayectorias, planificación de movimiento, configuración del espacio, muestreo del espacio, y varios algoritmos que utilizan dicho muestreo como base. De esta forma, y de acuerdo con la base teórica detallada en las secciones 2.2 y 2.3, se ha completado un estudio del estado del arte en planificación

de trayectorias y movimiento con un enfoque 3D detallado en el capítulo 6 [Samaniego, Sanchis, García-Nieto y Raul Simarro 2017]. Dicho estudio busca establecer la metodología más adecuada en planificación, que además se adecue a las características de movimiento del UAV de ala fija definido para el desarrollo de la tesis. Por otro lado, producto de tal estudio, en el capítulo 7 [Samaniego, Sanchis, García-Nieto y Raúl Simarro 2019] se presenta una evolución algorítmica en planificación desde un enfoque de descomposición de celdas adaptativo y dinámico.

En los capítulos 8 [Samaniego, Sanchís y col. 2018] y 9 [Samaniego, Sanchis, Garcia-Nieto y col. 2020] se han realizado un conjunto de simulaciones de vuelo, razón por la cual, en la sección 2.4 se ha revisado la temática de las restricciones no-holonómicas, además, se ha completado una breve representación matemática sobre vehículos aéreos (descripción que será ampliada en el capítulo 5, donde se detalla el modelo Cinemático y Dinámico de UAV utilizado en la tesis). Donde adicionalmente, se detalla la capacidad de movimiento libre y controlado de un UAV de ala fija, modelado como un cuerpo rígido.

# Capítulo 3

# Generación de trayectorias suaves

***Resumen:*** *El presente capítulo describe una formalización matemática y geométrica sobre la creación de curvas suaves, y la manera en la que se realiza una transición desde estas curvas hacia el problema de seguimiento de trayectorias suaves. Finalmente, se hace una breve descripción de una metodología muy utilizada y aplicada en la literatura sobre seguimiento de trayectorias llamada Pure-Puresuit.*

## 3.1 Introducción a las Curvas Algebráicas

Los orígenes del estudio matemático de las curvas se remontan a los trabajos de la antigua Grecia, y están ligados a los nombres de Pitágoras (c. 569 a. C. - c. 475 a. C.), Euclides (ca. 325 a. C. - ca. 265 a. C.) o Diofanto, Galois(1811 - 1832), matemáticos como Fermat(1601 - 1665), Euler (1707 - 1783), Lagrange (1736 - 1813) o Gauss (1777 - 1855). Estos y otros, plantearon y resolvieron cuestiones sobre un buen número de curvas, muchas de las cuales caen en la categoría de lo que en la actualidad se conocen como curvas algebraicas. Ejemplos de estas son las curvas cónicas, la curva cisoide de Diocles [Cordero, Fernandez y Gray 1995], la curva concoide de Nicomedes [Gómez y Torres 2006] etc.

Además de la circunferencia, que es sin duda la curva más conocida y utilizada, existen otros muchos tipos de curvas. En la actualidad, una parte importante de los objetos que se fabrican están construidos bajo algún tipo de forma de curva geométrica. Así, basta con prestar atención al entorno que nos rodea y concluir que en muchos de los objetos que nos rodean están presentes las curvas técnicas[1] y las curvas cónicas[2]. Por ejemplo, la forma de parábola que tienen algunos ojos de puente, o la forma de óvalo u ovoide con las que se han diseñado ciertas cucharas, etc. A continuación, se realizará una definición del concepto de curvas, seguido de un breve repaso de los tipos de curvas más conocidos y utilizados habitualmente.

## 3.2 Definición de Curvas Algebráicas

Realizar una definición de curva en un sentido general es una tarea compleja, pues en diferentes áreas de la geometría se describe de forma distinta.

**Definición 3.2.1** *En la geometría elemental y, en la geometría diferencial, una* ***curva*** *(o línea curva) es una línea continua de una dimensión, que varía de dirección paulatinamente. Entonces, esta curva geométrica se aparta constantemente de la dirección recta sin formar ángulos, y la trayectoria de los puntos que la forman es continua y, además, cumple una determinada norma.*

En geometría, una curva en el n-espacio euclídeo (según el teorema de la curva de Jordan [Hales 2007b; Hales 2007a]) es un conjunto $C \subset \mathbb{R}^n$ que es la imagen de un intervalo I abierto bajo una aplicación continua $\mathrm{xI} \to \mathbb{R}^n$, es decir:

$$\mathcal{C} = \{\mathrm{x}(t) \in \mathbb{R}^n : t \in \mathrm{I}\} \tag{3.1}$$

donde, $(\mathbf{x}, \mathrm{I})$ es una representación paramétrica o parametrización de $\mathcal{C}$. Entonces, una curva, en el plano o en el espacio tridimensional, es la imagen de un camino, que se considera con derivadas continuas a trozos en un intervalo de definición [Clapham 1992]. En este sentido, una curva puede ser expresada en diferentes formas canónicas, siendo estas:

[1] Son un conjunto de curvas que son muy utilizadas en ingeniería y en arquitectura, son sencillas de construir, ya que están formadas por arcos de distintas circunferencias unidos entre sí mediante tangencias.

[2] Cuando una recta $g$, que corta a otra recta $e$, gira en el espacio alrededor de ésta, se forma una superficie cónica. A la recta $g$ se le llama recta generatriz de la superficie cónica, a la recta $e$, eje del cono, y al punto de corte entre ambas, vértice $v$.

1) **En coordenadas cartesianas:**

a) En forma implícita:

$$F(x, y) = 0. \text{ Por ejemplo : } \sqrt{x^2 + y^2} = e^{xy} \tag{3.2}$$

b) En forma explícita:

$$y = f(x). \text{ Por ejemplo : } y = \frac{3x^2 + 5x - 7}{x^2} \tag{3.3}$$

c) En forma paramétrica:

$$\begin{aligned} &x = x(t), y = y(t). \\ &\text{Por ejemplo : } x = a \cdot \ln(t), y = \frac{a}{2}(2t + \frac{3}{t+1}) \end{aligned} \tag{3.4}$$

2) **En coordenadas paramétricas:**

$$\rho = f(\phi). \text{ Por ejemplo: } \rho = a \cdot \phi. \tag{3.5}$$

Un conjunto $\gamma$ de puntos del espacio se denominará curva elemental si es la imagen obtenida en el espacio por una aplicación topológica de un segmento abierto de recta.

Sea $\gamma$ una curva elemental y sea $a < t < b$ el segmento abierto del que se obtiene la aplicación $f$ de la curva correspondiente al punto $t$ del segmento. El sistema de igualdades:

$$x = f_1(t), y = f_2(t), z = f_3(t) \tag{3.6}$$

constituyen ecuaciones de la curva $\gamma$ en forma paramétrica.

En relación con esto, se pueden destacar varias clases de curvas como: la curva simple, curva plana, curva diferenciable, curva cerrada y la curva suave [Lafuente 1998]. Sobre este último tipo de curva se entrará en detalle a continuación en la sección 3.3.

## 3.3 Curvas suaves y aproximación en generación de trayectorias

En el análisis matemático, la suavidad de una función es una propiedad que se mide por el número de derivadas continuas que tiene sobre algún dominio [Warner 2013]. De esta forma, el mínimo requerimiento para que una función pueda considerarse "suave" es que debe ser diferenciable en todas partes (por lo tanto, continua [Porteous 2001]). Por otro lado, una curva suave también podría poseer derivadas de todos los órdenes de su dominio, en cuyo caso se denomina función C-infinito o $C^{\infty}$.

**Definición 3.3.1** *Se dice que una curva $C$, representada normalmente por $C(t)$ en un intervalo $I$, es suave, si $x'(t), y'(t), z'(t)$ son continuas en $I$ y no se anulan simultáneamente, excepto posiblemente en los puntos terminales de $I$. Se dice que la curva $C$ es suave a trozos si es suave en cada sub-intervalo de alguna partición de $I$.*

Si se asume que el resultado de una planificación de trayectorias genera un conjunto puntos en el espacio euclídeo, entonces sobre estos se puede aplicar un control de seguimiento de los puntos de la trayectoria con fines de orientación. No obstante, una mejor opción de implementación del control de orientación y navegación es crear una trayectoria suave, que considere las limitaciones dinámicas de los AVs. Los puntos de la trayectoria pueden conectarse para generar segmentos de ruta suaves, lo que preserva la continuidad de la curvatura entre diferentes segmentos que podrían ser líneas y/o arcos de circunferencias, al tiempo que se minimiza la curvatura máxima de la curva [Schouwenaars, How y Feron 2004].

En la Figura 3.1, se puede ver un ejemplo de trayectoria suave y continua (línea punteada roja) para la navegación de un UAV. La trayectoria evita giros "bruscos", de tal modo que el UAV pueda realizar maniobras sin detenerse.

Algunos planificadores de trayectoria no consideran la generación de una ruta suave navegable por un AV, limitándose a una generación de trayectorias en función del tiempo [W. Cai, M. Zhang e Y. R. Zheng 2017]. Esto significa que la trayectoria planificada podría no garantizar una perfecta evasión de obstáculos debido a la posibilidad de la diferencia entre la trayectoria planificada y la trayectoria real ejecutada. En consecuencia, se requiere un suavizado adicional para evitar tales problemas, y aun así es difícil asegurar que el camino resultante cumpla con las restricciones cinemáticas en las tres dimensiones simultáneamente. Por lo tanto, las restricciones a menudo se calculan secuencialmente

de dos en dos dimensiones (es decir sobre el par de dimensiones desacopladas), lo que limita la capacidad de optimización, pues optimizar el vuelo horizontal no optimiza el vuelo vertical y viceversa, lo que indica que una mejor opción es realizar una optimización completa de vuelo 3D.

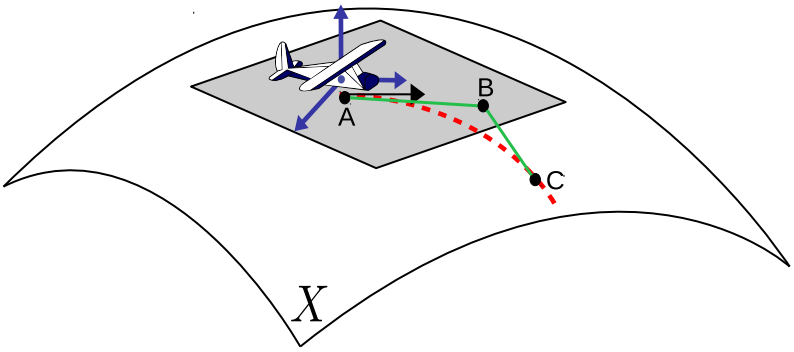


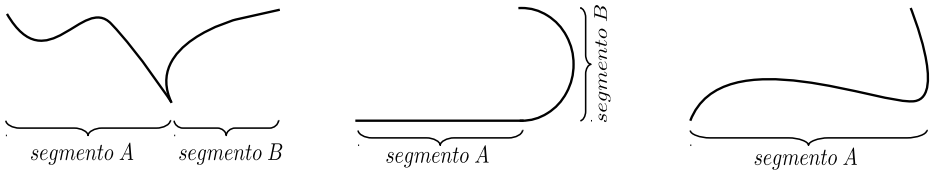

**Figura 3.1:** La línea de color verde indica la trayectoria directa entre los puntos $A, B$ y $C$, mientras que la línea punteada de color rojo muestra un camino continuo y suave.

Una aproximación sencilla para la generación de trayectorias es utilizar curvas que concatenen segmentos de líneas y círculos [Dubins 1957; Jacobs, Heinzinger y col. 1989; Reeds y Shepp 1990]. En este sentido, se puede definir la suavidad de una curva en función de su continuidad (siendo la continuidad geométrica o paramétrica). Del mismo modo, existen diferentes curvas que son objeto de estudio y que hacen una transición entre rectas y arcos. Por ejemplo, las curvas cúbicas [Hartman, Y. Kanayama y Smith 1989], clotoides [Shin, Singh y Whittaker 1992], splines [Delingette, Hebert e Ikeuchi 1991], curvas Bézier [Klančar y Škrjanc 2010], etc.

**Definición 3.3.2** *Se dice que una* ***curva es de clase*** $G^n$ *(tiene continuidad geométrica de orden* $n$*, donde* $n \geq 1$*) si existe una parametrización regular local de clase* $C^n$ *de esta curva en una vecindad de cada punto de esa curva [Kiciak 2016].*

**Definición 3.3.3** *Se dice que una* ***superficie es de clase*** $G^n$ *(tiene continuidad geométrica de orden* $n$*, donde* $n \geq 1$*) si existe una parametrización regular local de clase* $C^n$ *de esta superficie en una vecindad de cada punto de esa superficie [Kiciak 2016].*

En consecuencia, la continuidad geométrica asegura que los puntos finales de los distintos segmentos de una curva se encuentren y que además las direcciones del vector tangente sean iguales [A. Ravankar, Ankit A. Ravankar y col. 2018]. Mientras que la continuidad paramétrica asegura que los puntos finales de los diversos segmentos del camino se encuentren, y que además la dirección y las magnitudes del vector tangente sean iguales. Finalmente, es importante resaltar que, la continuidad paramétrica ($C^i$) implica la continuidad geométrica ($G^i$) es decir $C^i \Rightarrow G^i$, sin embargo, no a la inversa $G^i \nRightarrow C^i$.




(**a**)Discontinuidad tangencial.

(**b**)Discontinuidad en curvatura.

(**c**)Continuidad en curvatura.

**Figura 3.2:** Continuidad paramétrica.

La Figura 3.2 muestra diferentes clases de continuidades paramétricas. La Figura 3.2(a) presenta una continuidad de clase $C^0$ (continuidad tangencial), donde los puntos del camino se conectan con todos los puntos entre el inicio y el final, no obstante, existe una discontinuidad tangencial en los puntos de unión entre los segmentos $A$ y $B$. La Figura 3.2(b) muestra una continuidad de clase $C^1$, a su vez contiene una continuidad $C^0$ al preservar el punto de tangencia en los puntos de unión entre segmentos $A$ y $B$. Es importante resaltar que el camino pasa por un segmento $A$ con una curvatura cero hasta el segmento $B$ con un valor de curvatura finito. La Figura 3.2(c) muestra un camino de clase $C^2$, este camino conserva valores diferenciales de segundo orden en cada punto. Por lo tanto, el camino $C^1$ es más suave que $C^0$ en los puntos de tangencia que unen los segmentos, mientras que $C^2$ es más suave que $C^1$ en los puntos de unión de sus segmentos.

Como se menciona en [A. Ravankar, Ankit A. Ravankar y col. 2018], desde el punto de vista de la planificación de trayectorias y de movimiento, los términos de continuidad paramétrica $C^i$ significan suavidad tanto de la curva como de su parametrización, en tanto que continuidad geométrica $G^i$ significa la suavidad de la trayectoria realizada por el AV [G. E. Farin y G. Farin 2002]. Con respecto a esto, continuidad $C^1$ significa continuidad del vector de la tangente, mientras que la continuidad $G^1$ significa continuidad de la pendiente. Por otra parte,

continuidad $C^2$ significa continuidad del vector aceleración, mientras que la continuidad $G^2$ significa continuidad de la curvatura. En cuanto al movimiento del AV, el movimiento continuo $C^1$ conserva la velocidad, mientras que la trayectoria continua $C^2$ conserva la aceleración. Finalmente, en el campo de la planificación de trayectorias en AVs, el enfoque de continuidad aceptable es de tipo $C^1$ o $C^2$.

Aunque, una continuidad de clase $C^1$ es adecuada para AVs que navegan a velocidades bajas, para mayores velocidades y aceleraciones es necesario una transición entre líneas rectas y curvas [A. Ravankar, Ankit A Ravankar y col. 2016]. De esta manera, al conectar una línea recta y una curva, los valores de las curvaturas cambian de valor cero a un valor finito (con valores de tasa uniformes). Finalmente, es importante destacar que las curvas de transición presentan propiedades importantes para el movimiento de los AVs, siendo: a) Son tangenciales a la recta, es decir, la curvatura al comienzo es cero. b) Se unen a la curva circular de forma tangencial. c) Su curvatura aumenta a la misma velocidad.

## 3.4 Aproximación a generación de trayectorias

Un AV debe realizar el seguimiento de una trayectoria sin detenerse, por lo tanto, un AV requiere de una curva de transición que una puntos con diferentes valores de curvatura. Estas curvas de transición garantizan la generación y factibilidad de trayectorias continuas para los vehículos no-holonómicos.

En la literatura se mencionan varias clases de curvas que pueden ser aproximadas y tratadas con el objetivo de que, un AV pueda navegar por ellas. En esta sección se hará un breve resumen de curvas utilizadas para la navegación.

En distintos trabajos se aborda esta temática como [Boissonnat, Cérézo y Leblond 1991; Soueres y Laumond 1996] demostrando que esta clase de trayectorias son óptimas, y aunque puedan parecer continuas en un plano cartesiano (con continuidad $C^1$), la curvatura resultante es discontinua (sin continuidad $C^2$), lo cual indica que una continuidad de tipo $C^1$ es admisible en navegación AV.

Existen diferentes curvas que han sido utilizadas con el objetivo de servir como curvas de transición entre arcos circulares y líneas rectas, siendo ejemplos de estas curvas: las espirales cúbicas [Y. J. Kanayama y Hartman 1997], clotoides [Y. Kanayama 1986; Shin, Singh y Whittaker 1990], splines [Delingette, Hebert

e Ikeuchi 1991], además de otras trayectorias polinómicas [Segovia y col. 1991; Nagy y Kelly 2001].

### *3.4.1 Trayectorias basadas en Clotoides*

Una extendida aproximación para la generación de trayectorias utiliza curvas que concatenen segmentos de líneas y círculos [Jacobs y Canny 1993; Reeds y Shepp 1990]. Estas curvas de tipo clotoide, han demostrado que, trayectorias más cortas conectan dos puntos del plano euclídeo con una curvatura limitada.

La clotoide [Weisstein 2016a], también conocida como espiral de Euler o espiral de Cornu (ver Figura 3.3), es la curva parametrizada en el plano complejo definida por puntos, siendo:

$$B(t) = S(t) + iC(t) \tag{3.7}$$

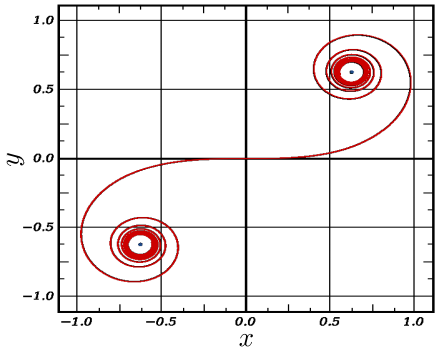
donde,

$$\begin{aligned} x(t) = C(t) &= \int_0^t \cos\left(\frac{\pi}{2}s^2\right) ds \\ y(t) = S(t) &= \int_0^t \sin\left(\frac{\pi}{2}s^2\right) ds \end{aligned} \tag{3.8}$$

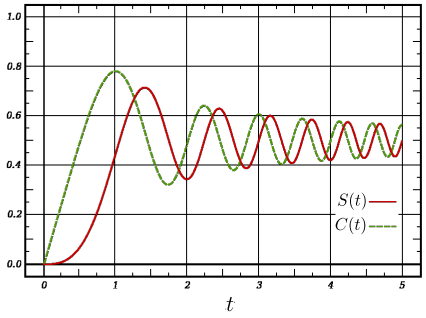
donde $C$ y $S$ son las funciones de Fresnel [Weisstein 2016b], que también suelen ser llamadas el coseno y el seno integral de Fresnel.

Ambas funciones de Fresnel se aproximan a $\frac{1}{2}$ cuando $t \to \infty$ y así la curva gira lentamente hacia $\left(\frac{1}{2}, \frac{1}{2}\right)$ en el primer cuadrante. Y debido a la simetría, ya que ambas funciones son impares, la curva gira hacia $\left(-\frac{1}{2}, -\frac{1}{2}\right)$ en el tercer cuadrante. La curva de Cornú tiene una propiedad relevante, la cual indica que su curvatura es proporcional a la distancia a lo largo de la trayectoria de la curva. Por lo tanto, un AV que viaja a velocidad constante experimentará una tasa constante de aceleración angular mientras viaja alrededor de la curva, lo que significa que el giro se ejecuta a una tasa constante, y en consecuencia se realiza un giro más seguro [A. Ravankar, Ankit A. Ravankar y col. 2018].

En este sentido, las clotoides poseen interesantes propiedades geométricas, debido a su estrecha relación entre los fenómenos físicos (velocidad, aceleración, etc.) mediante el parámetro de escalado de los clotoides, lo cual implica un beneficio en cuanto a comodidad y seguridad [Girbés Juan 2016].




(**a**)Espiral de Cornú $(x, y) = (C(t), S(t))$. La espiral converge en el centro de los dos remolinos extremos de la imagen, a medida que $t$ tiende a más infinito y menos infinito.




(**b**)Integrales normalizadas de Fresnel, $S(t)$ y $C(t)$.

**Figura 3.3:** Espiral de Cornu, o clotoide: Fuente [Graham 2002].

Las propiedades descritas hacen que los coángulos[3] sean una opción interesante para suavizar las trayectorias. En [Gim y col. 2017], se ha propuesto el suavizado de la trayectoria para la navegación de AVs de ruedas, donde se genera una trayectoria de curvatura continua mediante un ajuste paramétrico de múltiples clotoides.

### *3.4.2 Trayectorias basadas en la función B-spline*

Una *B-spline* (*Basis spline* del inglés *Línea Polinómica Suave Básica*), perteneciente al subcampo matemático de análisis numérico, es una función *spline* (una secuencia de segmentos de curva que se conectan entre sí para formar una sola curva continua) [De Boor y col. 1978] que tiene el mínimo soporte con respecto a un determinado grado, suavidad y partición del dominio. Mientras que, en el subcampo de la informática de diseño asistido por computador, el término B-spline se refiere con frecuencia a una curva parametrizada por otras funciones spline, que se expresan como combinaciones lineales de B-splines.

La función spline unidimensional $b$ puede ser expresada como:

[3]Son ángulos en posición estándar (ángulos con el lado inicial en el eje positivo de las $x$) que tienen un lado terminal común.

$$b(u) = \sum_{j=0}^{m} b_j N_j^d(u) \tag{3.9}$$

donde $b_j$ son puntos de control y $N_j^d(u)$ son las funciones de la B-spline de grado $d$ que son definidas sobre una secuencia de nudos no decrecientes $u_k$ tal que $u_0 \leq u_1 \leq, \ldots, \leq u_{m+d+1}$. El número de nudos está determinado por la suma del número de puntos de control $(m+1)$ y el orden de la B-spline $(d+1)$. El primer y el último nudo de la secuencia deben tener una multiplicidad $(d+1)$ para que la B-spline pase a través del primer y el último punto de control, de tal manera que $u_0 = u_1 = \cdots = u_d$ y $u_{m+1} = u_{m+2} = \cdots = u_{m+d+1}$, respectivamente.

Las funciones de la B-spline se calculan mediante las fórmulas de Cox-deBoor [De Boor y col. 1978] desde el grado 0 hasta el grado $d$ de la siguiente manera:

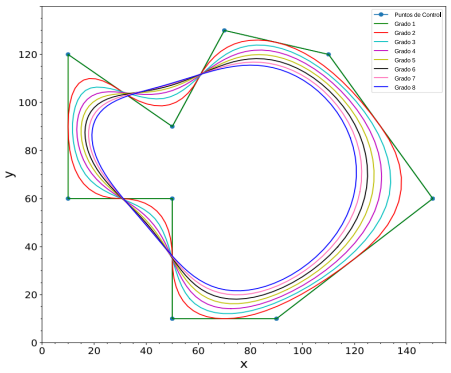
$$N_j^0(u) = \begin{cases} 1 & if\ u_j \leq u \leq u_{j+1}, \\ 0 & \text{de lo contrario,} \end{cases} \tag{3.10}$$

$$N_j^d(u) = \frac{u - u_j}{u_{j+d} - u_j} N_j^{d-1}(u) + \frac{u_{j+d+1} - u}{u_{j+d+1} - u_{j+1}} N_{j+1}^{d-1}(u) \tag{3.11}$$

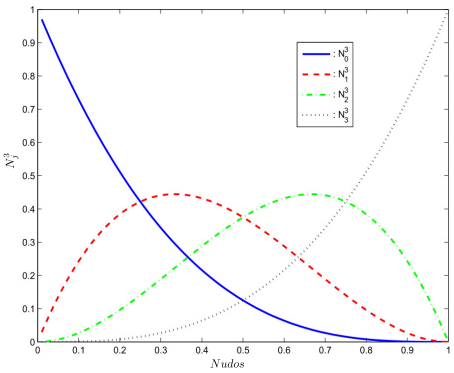
Cuando los nudos son equidistantes, se dice que la B-spline es *uniforme*, de otro modo sería *no uniforme* (Nótese que $j+d+1$ no puede exceder de $m-1$, lo que limita tanto a $j$ como a $d$). Mientras que, si dos nudos $u_j$ son idénticos, cualquiera de las posibles formas indeterminadas $0/0$ se consideran 0.

La función B-spline tiene varias propiedades importantes [Piegl y Tiller 2012], entre las cuales se destacan: 1) Su soporte local, es decir, cada vértice afecta a la forma de la curva en un intervalo definido. 2) Una B-spline pasa por el primer y el último punto de control. 3) La posibilidad de cambio en el orden, sin cambiar el número de vértices del polígono de control. 4) Una B-spline de orden 3 es tangente a la parte media de los lados del polígono de control.

En relación con esto, la B-spline es una herramienta interesante para suavizar trayectorias, ya que se puede generar trayectorias para diferentes grados y configuraciones sobre el mismo conjunto de puntos de control (ver Figura 3.4(a)), manteniendo los puntos inicio $q_I$ y final $q_G$. En este tipo de configuración, las curvas generadas pueden estar demasiado lejos de los puntos reales. Sin embargo, al duplicar el número de puntos de control mediante la introducción de puntos de control adicionales entre dos puntos de control originales, las curvas se suavizan, del mismo modo que las trayectorias.




(**a**)B-spline con 9 puntos de Control en una trayectoria cerrada.




(**b**)B-spline $N_j^3$ cúbica de grado 3 sobre una secuencia de nudos $u \in (0, 1)$.

**Figura 3.4:** Curva suave de tipo B-spline Fuente: [A. Ravankar, Ankit A. Ravankar y col. 2018].

## 3.5 Seguimiento de la trayectoria

La orientación y el control de la navegación de los AVs es un tema de investigación importante en la actualidad. En particular, en el caso de los UAVs, la capacidad de control de orientación y navegación totalmente automatizada permite a los UAVs cumplir misiones en diversas circunstancias con una mínima intervención humana.

Un enfoque clásico es construir un camino que converja en una línea virtual dibujada sobre una trayectoria, es decir, en base a datos directos generados por un sistema de control basado, por ejemplo, en la visión artificial. Otra aplicación clásica es estimar el perfil de trayectoria (definición de un conjunto de movimientos realizables por un UAV) y proporcionar una ley de control que mantenga al UAV dentro de los límites de la trayectoria.

La metodología *Seguimiento-Puro* (del inglés *Pure-Pursuit*) determina valores apropiados de velocidad que además garantiza una convergencia hacia una determinada trayectoria basada en la postura actual del AV [Wallace y col. 1985; Ollero Baturone 2001]. En este tipo de metodologías se busca calcular las configuraciones de los puntos objetivos en base a la posición actual del AV con relación a la trayectoria, encontrando un punto que tenga una separación mínima de distancia (ver Figura 3.5). En consecuencia, este proceso se puede

aplicar de manera iterativa, logrando así mantener la velocidad del AV en función de su posición. Otros enfoques, incluyen leyes de control lineales o no lineales basadas en el modelo del AV, lo cual garantiza la convergencia [Ollero y Amidi 1991; Campion, Bastin y Dandrea-Novel 1996].

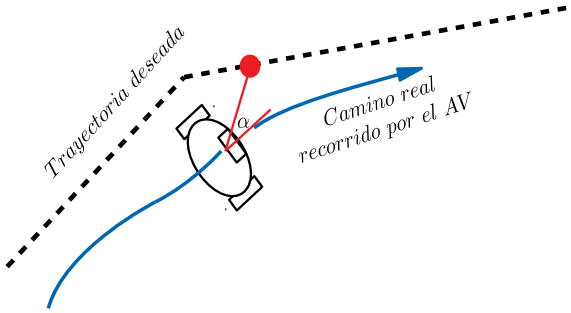



**Figura 3.5:** La línea punteada negra representa la trayectoria a seguir, mientras que la línea azul muestra la trayectoria real seguida por el AV. La metodología de Seguimiento-Puro consiste en realizar los cálculos geométricos necesarios con el fin de conectar el centro de la ubicación del eje del AV con el punto objetivo en la trayectoria.

Finalmente, dentro del contexto de seguimiento de trayectorias y control cinemático AV, las metodologías de planificación de movimientos difieren debido a las configuraciones cinemáticas particulares de los AVs. Por lo tanto, para generar soluciones factibles, es necesario desarrollar un instrumento capaz de generar referencias de curvas suaves y continuas para las aplicaciones de planificación de trayectorias y evasión de obstáculos.

## 3.6 Conclusiones del capítulo

En este capítulo se ha descrito una base teórica sobre las definiciones de curvas suaves, y su transición hacia trayectorias suaves. Así mismo, se ha mencionado diversas metodologías de construcción de curvas que realizan una buena aproximación hacia la generación de trayectorias suaves, como se puede ver en el estudio realizado en el capítulo 8 [Samaniego, Sanchís y col. 2018] y 9 [Samaniego, Sanchis, Garcia-Nieto y col. 2020]. Por otra parte, sin importar la trayectoria suave generada, es necesario precisar una metodología de seguimiento de tales trayectorias, por lo que el capítulo finaliza con una breve descripción de la metodología de seguimiento de trayectoria (sección 3.5) adoptada en el trabajo de tesis, llamada Seguimiento-Puro.

De la misma manera, se han descrito técnicas de concatenación de segmentos de línea recta y arcos circulares como las B-spline o clotoides (en la sección 3.4), que funcionan como curvas de transición a fin de garantizar una curva continua.

En el capítulo 9 [Samaniego, Sanchis, Garcia-Nieto y col. 2020] se lleva a cabo una aplicación en la que se utiliza una trayectoria en base a segmentos de línea recta y arcos esféricos, tratados como un problema de Optimización Multiobjetivo (base teórica descrita en el capítulo 4), que además toma en consideración la evasión de obstáculos y seguimiento óptimo de la trayectoria.

# Capítulo 4

# Trayectorias suaves utilizando optimización multiobjetivo

***Resumen:*** *En este capítulo se hará un breve resumen sobre la resolución de los problemas de optimización, aplicando la metodología del Problema de Optimización Multiobjetivo (MOP). A lo largo de este capítulo se busca definir el proceso que sigue el MOP, sus elementos principales, y finalmente sus pasos fundamentales.*

## 4.1 Introducción a la optimización multiobjetivo

En matemáticas, estadísticas, ciencias empíricas, ciencia de la computación, etc; la optimización matemática (optimización o programación matemática) significa la selección del mejor elemento (con respecto a algún criterio) de un conjunto de elementos disponibles.

La optimización es fundamental para cualquier problema que implique la toma de decisiones, ya sea en la ingeniería o en la economía. Entonces, la tarea de la toma de decisiones implica elegir entre varias alternativas. Esta elección se rige por la necesidad de tomar la "mejor" decisión. De esta manera, la medida de la bondad de las alternativas se describe mediante una función objetivo o un índice de rendimiento. La teoría y los métodos de optimización se ocupan de la selección de la mejor alternativa en base a la definición de una función objetivo. En el caso más simple, un problema de optimización consiste en

maximizar o minimizar una función real eligiendo sistemáticamente valores de entrada (tomados de un conjunto permitido) y computando el valor de la función.

En el campo de las ciencias e ingeniería, habitualmente se presentan problemas que requieren la optimización simultánea de más de un objetivo (Optimización Multiobjetivo). Por lo tanto, se puede completar el diseño de ingeniería como un *Problema de Optimización Multiobjetivo* (MOP, acrónimo del inglés *Multi-objective optimization problem*), para lo cual, primero se debe definir el problema de diseño, lo que implica identificar las variables de decisión $\theta$ y los objetivos de diseño $J = [J_1(\theta), \ldots, J_m(\theta)]$. Si $m = 1$, entonces se dice que se trata de un problema de un solo objetivo, mientras que si $m \geq 2$ se define como un MOP.

De manera regular, se utilizan dos enfoques para resolver un enunciado de optimización para un MOP: 1) El enfoque de la *Función Objetivo Agregado* (AOF, del acrónimo *Aggregate Objective Function*), y 2) El enfoque de *Generar Primero Elegir Después* (GFCL, del acrónimo *Generate First, Choose Later*) [Mattson y Messac 2005].

En el contexto de AOF se define un enunciado de optimización de índice único que fusiona los objetivos de diseño. Un ejemplo tradicional consiste en utilizar un vector de ponderación para indicar la importancia relativa entre los objetivos. En tales casos, el responsable de la *Toma de Decisiones* (DM, del acrónimo *Decision Maker* o simplemente el diseñador) necesita describir todos los factores de ponderación o pesos a la vez, al principio del proceso de optimización.

En el enfoque de la GFCL, el objetivo principal es generar muchas soluciones óptimas potencialmente deseables, y luego seleccionar la más adecuada. Esto se debe a la imposibilidad de obtener una solución que sea la mejor para todos los objetivos, por lo que pueden aparecer varias soluciones con diferentes niveles de compensación. La selección tiene lugar en un paso de *Toma de Decisiones Multicriterio* (MCDM, del acrónimo *Multicriteria Decision Making*), en el que la tarea del DM es analizar el equilibrio entre los objetivos y seleccionar la mejor solución según sus preferencias. La fase de DM hace uso de distintas técnicas [Clímaco y Craveirinha 2005] y enfoques de visualización [Lotov y Kaisa Miettinen 2008], y no es una tarea trivial ya que la complejidad del análisis y la visualización aumentan con el número de objetivos. Además, este proceso podría consumir más tiempo que el propio proceso de optimización [C. A. C. Coello, Lamont, Van Veldhuizen y col. 2007].

Una forma de generar tales conjuntos de soluciones potenciales en el enfoque GFCL es usar algoritmos de optimización multiobjetivo. Este enfoque de optimización busca un conjunto de soluciones óptimas para aproximarse a lo que se conoce como el conjunto de Pareto [KM Miettinen 1999]. Una aproximación al conjunto de Pareto proporciona una idea preliminar del espacio objetivo que además es muy útil cuando se necesita explicar y justificar el procedimiento MCDM. No obstante, su mayor desventaja se enfoca sobre su requerimiento de tiempo e incorporación del DM en el proceso general.

La metodología de optimización multiobjetivo para aproximar conjuntos de soluciones óptimas ha sido utilizada en numerosas áreas del conocimiento y aplicaciones [C. A. C. Coello y Lamont 2004], siendo ejemplos de estos: cadenas de suministro [Trisna y col. 2016; Mansouri, Gallear y Askariazad 2012], sistemas de energía [Fadaee y Radzi 2012], flujo de programación en masa [Y. Sun y col. 2011], reconocimiento de patrones [C. A. C. Coello 2011], modelado hidrológico [Efstratiadis y Koutsoyiannis 2010], recursos hídricos [Reed y col. 2013] y la gestión de la cartera [Metaxiotis y Liagkouras 2012].

## 4.2 Problema de optimización multiobjetivo MOP

La optimización multiobjetivo es ciertamente un área crucial en la ingeniería y la ciencia. De hecho, innumerables problemas de optimización en la vida real se caracterizan por sus múltiples objetivos. Uno de los enfoques más empleados hoy en día para abordar problemas complejos multiobjetivo es el uso de metaheurísticas[1]. Donde el resultado es generalmente un conjunto de soluciones óptimas de Pareto en lugar de una solución única como en la optimización mono-objetivo. De hecho, se han realizado diferentes investigaciones sobre este campo, siendo algunas de estas alternativas: los algoritmos evolutivos, como las colonias de hormigas [García-Martínez, Cordón y Herrera 2007], la optimización por enjambres de partículas [Reyes-Sierra, C. C. Coello y col. 2006], la colonia de abejas [Pham y Ghanbarzadeh 2007; Sayadi y col. 2009], las evoluciones diferenciales [Xue, Sanderson y Graves 2005], y los sistemas inmunes artificiales [C. A. C. Coello y N. C. Cortés 2005].

[1]Una metaheurística es un método heurístico (una técnica, método o procedimiento inteligente que busca realizar una tarea que no es producto de un riguroso análisis formal, sino de conocimiento experto sobre la tarea) para resolver un tipo de problema computacional general, usando los parámetros dados sobre procedimientos genéricos y abstractos. Generalmente se aplican a problemas que no tienen un algoritmo o heurística específica que dé una solución satisfactoria; o bien cuando no es posible implementar ese método óptimo. La mayoría de las metaheurísticas tienen como objetivo los problemas de optimización combinatoria.

Además de los componentes de búsqueda clásicos de la metaheurística mono-objetivo (por ejemplo: representación, manejo de restricciones, operadores de búsqueda), la metaheurística multiobjetivo debe manejar tres componentes de búsqueda: 1) Asignación de aptitud. 2) Preservación de la diversidad. 3) Elitismo; un enfoque de la metodología general se puede ver en la Figura 4.1.

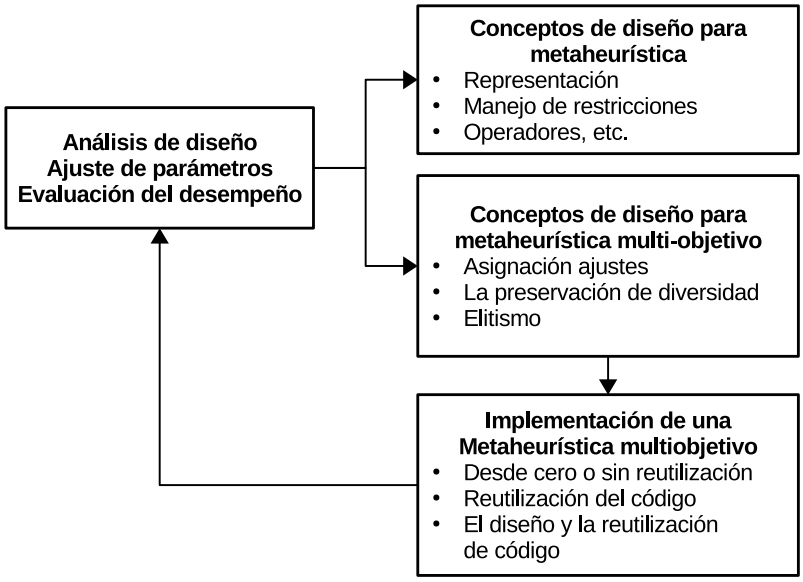



**Figura 4.1:** El proceso de desarrollo base multiobjetivo [Talbi 2009].

Al igual que en la optimización mono-objetivo, el problema de optimización multiobjetivo MOP puede dividirse en dos categorías: 1) Cuando las soluciones están codificadas con variables de valor real (también conocidas como *MOP continuas*). 2) Aquellas en las que las soluciones se codifican utilizando variables discretas (también conocidas como *MOP combinatorias*).

En la clase de MOP continua, se puede destacar los MOP continuos complejos (por ejemplo, si la formulación del problema contiene al menos una función no lineal) para las que no se pueden aplicar algoritmos exactos en un tiempo razonable. Un número infinito de soluciones óptimas de Pareto pueden componer el frente de Pareto de una MOP continua, mientras que en las MOP combinatorias, tanto el conjunto de soluciones factibles $D$ como el conjunto de Pareto son finitos.

**Definición 4.2.1** *Un **Problema de optimización multiobjetivo MOP** puede definirse como el problema de encontrar un vector de variables de decisión que satisfagan un cierto conjunto de restricciones y optimice un conjunto de funciones objetivo [Osyczka 1985].*

Un problema de optimización puede ser representado por las siguientes ecuaciones [S. Boyd, S. P. Boyd y Vandenberghe 2004]:

$$\begin{aligned} \min_{\theta} \mathbf{J}(\boldsymbol{\theta}) =& [J_1(\boldsymbol{\theta}), J_2(\boldsymbol{\theta}), \ldots, J_n(\boldsymbol{\theta})] \\ & \boldsymbol{\theta}_{min} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_{max} \end{aligned} \tag{4.1}$$

sujeto a:

$$g_i(\boldsymbol{\theta}) \quad \leq 0, i = 1, \ldots, m \tag{4.2}$$

$$h_j(\boldsymbol{\theta}) \quad = 0, j = 1, \ldots, p \tag{4.3}$$

Las ecuaciones anteriores, describen el problema de encontrar un $\boldsymbol{\theta}$ que minimice $\mathbf{J}(\boldsymbol{\theta})$ entre todas las $\boldsymbol{\theta}$ que satisfacen las condiciones $g_i(\boldsymbol{\theta}) \leq 0, i = 1, \ldots, m$, y $h_j(\boldsymbol{\theta}) = 0, j = 1, \ldots, p$.

Donde, los $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ se conocen como las variable de decisión y $\mathbf{J}(\boldsymbol{\theta}) : \mathbb{R}^{n_\theta} \to \mathbb{R}$ es la función objetivo (comúnmente llamada índice de coste). Las desigualdades $g_i(\boldsymbol{\theta}) \leq 0$, $g_i : \mathbb{R}^{n_\theta} \to \mathbb{R}$ representan las restricciones de desigualdad, mientras que $h_j(\boldsymbol{\theta}) = 0$, $h_j : \mathbb{R}^{n_\theta} \to \mathbb{R}$ son las restricciones de igualdad.

El conjunto de puntos para los que se definen la función objetivo y todas las restricciones:

$$D = \bigcap_{i=0}^{m} \text{dom } g_i \cap \bigcap_{j=0}^{p} \text{dom } h_j \tag{4.4}$$

representa el conjunto de soluciones factibles asociadas a las limitaciones de igualdad y desigualdad, y a los límites explícitos, que también se conoce como dominio del problema de optimización definido en la ecuación 4.1. Entonces, un punto $\boldsymbol{\theta} \in D$ es factible si satisface las restricciones 4.2 y 4.3.

En la Figura 4.2 se puede ver el espacio de soluciones factibles $D$ a la izquierda (de dimensión 2), mientras que el espacio objetivo a la derecha (de dimensión 3). Una función de costo del espacio de decisión en el espacio objetivo

evalúa la calidad de cada solución $(\theta_1, \ldots, \theta_k)$ asignando a un vector objetivo $(J_1, \ldots, J_n)$, que representa la calidad de la solución. En el campo de la optimización multiobjetivo, el DM la utiliza para trabajar en términos de evaluación de una solución en cada criterio, y se coloca en el espacio objetivo.

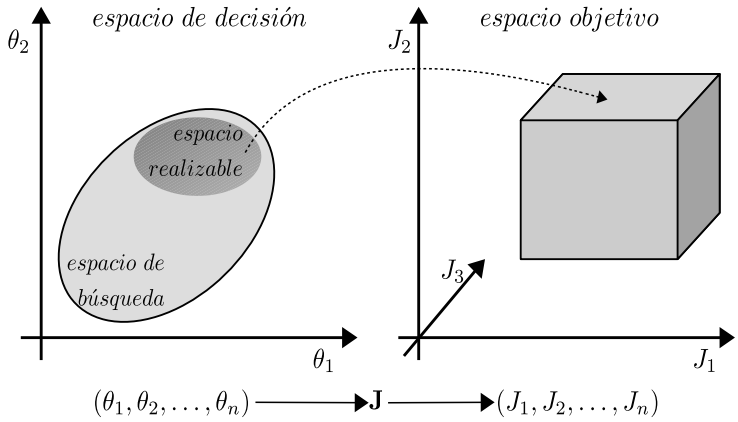



**Figura 4.2:** Ejemplo de espacio de decisión (2D) y espacio Objetivo (3D) en un MOP.

Por lo general, no es posible encontrar una solución que satisfaga todos los requisitos al mismo tiempo, por lo que una optimización proporciona un número de soluciones que no son mejores que ninguna otra en todos los objetivos al mismo tiempo. La noción más aceptada de óptimo en el entorno de problemas multiobjetivo es la propuesta originalmente por Francis Ysidro Edgeworth en 1881 y generalizada posteriormente por Vilfredo Pareto en 1896 [Ehrgott 2012].

**Definición 4.2.2** *Se dice que un punto $\boldsymbol{\theta}^* \in D$ es un **óptimo de Pareto** si para toda $\boldsymbol{\theta} \in D$ e $I = \{1, 2, \ldots, k\}$ ya sea:*

$$\forall_{i \in I}(J_i(\boldsymbol{\theta}) = J_i(\boldsymbol{\theta}^*)) \tag{4.5}$$

*o que al menos una $i \in I$ tal que:*

$$J_i(\boldsymbol{\theta}) > J_i(\boldsymbol{\theta}^*) \tag{4.6}$$

**Definición 4.2.3** *Se dice **dominancia de Pareto** cuando un vector $\overrightarrow{u} = (u_1, \ldots, u_k)$ domina a otro $\overrightarrow{v} = (v_1, \ldots, v_k)$ (denotado mediante $\overrightarrow{u} \preceq \overrightarrow{v}$) si*

*y solo si u es parcialmente menor a v, i.e.* $\forall i \in \{1,\ldots,k\}, u_i \leq v_i \wedge \exists i \in \{1,\ldots,k\} : u_i < v_i$.

**Definición 4.2.4** *Para un problema multiobjetivo dado, el* ***conjunto de óptimos de Pareto*** $(P^*)$ *se define como:*

$$P^* = \{\boldsymbol{\theta} \in D | \neg\exists \boldsymbol{\theta}' \in DJ(\boldsymbol{\theta}') \preceq J(\boldsymbol{\theta})\} \tag{4.7}$$

**Definición 4.2.5** *Para un problema multiobjetivo dado y un conjunto de óptimos de Pareto* $P^*$*, el* ***frente de Pareto*** $(PF^*)$ *se define como:*

$$PF^* = \{\overrightarrow{u} = J(\boldsymbol{\theta}) = (J_1(\boldsymbol{\theta}),\ldots,J_k(\boldsymbol{\theta}) | \boldsymbol{\theta} \in P^*\} \tag{4.8}$$

Por lo tanto, solucionar una optimización multiobjetivo es determinar el conjunto de todas las soluciones no dominadas (de mejor rendimiento), lo que se conoce como conjunto óptimo de Pareto $P^*$. La curva de este conjunto en el espacio objetivo se denota como el frente de Pareto $PF^*$, descrito en la Figura 4.3. El frente de Pareto $PF^*$ constituye la curva de compensación entre los objetivos en conflicto (la solución del MOP determina una curva de equilibrio que relaciona las funciones $J_1(\boldsymbol{\theta})$ y $J_2(\boldsymbol{\theta})$ para el ejemplo).

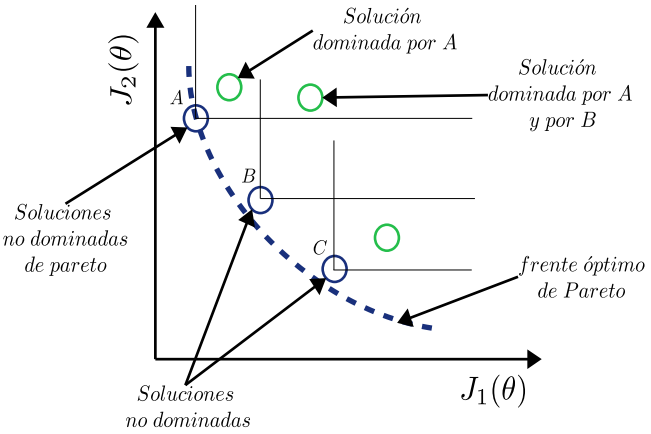



**Figura 4.3:** Ejemplo de Soluciones dominadas y soluciones de Pareto. Las soluciones de Pareto dominan el resto de las soluciones con respecto a la minimización de los objetivos $J_1(\boldsymbol{\theta})$ y $J_2(\boldsymbol{\theta})$.

En este sentido, en el ejemplo de la Figura 4.3, se muestran seis soluciones diferentes (círculos azules y verdes) que se aproximan a un frente de Pareto (línea azul punteada). Las soluciones $A$, $B$ y $C$ son soluciones no dominadas, ya que no hay mejores vectores de solución (en el conjunto calculado) para todos los objetivos. Las soluciones $B$ y $C$ no son óptimas de Pareto, ya que algunas soluciones (no encontradas en este caso) las dominan. Además, la solución $A$ también es óptima de Pareto, ya que se encuentra en el frente de Pareto $PF^*$. El conjunto de soluciones no dominadas ($A, B$ y $C$) construye el frente de Pareto $PF^*$ y el conjunto de Pareto $P^*$. Es importante notar que, la mayoría de las veces el frente de Pareto $PF^*$ es desconocido y sólo se basa en aproximaciones.

Con el objetivo de realizar un refinamiento en el frente de Pareto y de diferenciar los conceptos de diseño, sobre el frente de Pareto se define un concepto de diseño [Mattson y Messac 2005], que es una idea sobre la resolución del MOP. Este concepto de diseño se construye con un conjunto de soluciones óptimas de Pareto que son soluciones específicas en el concepto de diseño.

En la Figura 4.4 se puede apreciar un ejemplo de una aproximación al frente de Pareto (línea en negrita), para un concepto de diseño particular (círculos azules). Sin embargo, a menudo suelen haber diferentes conceptos, todos los cuales son viables para resolver un MOP. Por lo tanto, el DM puede calcular una aproximación del frente de Pareto para cada uno de los conceptos con el fin de hacer una comparación. En consecuencia, en [Mattson y Messac 2005] la definición de un frente de Pareto y de la optimización de Pareto se extendió a un frente de Pareto para un conjunto de conceptos (frente s-Pareto) donde todas las soluciones son s-Pareto óptimas.

**Definición 4.2.6** *Dado un MOP y $K$ diseños de conceptos, el vector objetivo $J_i(\boldsymbol{\theta}^1)$ es* ***s-Pareto óptimo*** *si no hay otro vector objetivo $J_i(\boldsymbol{\theta}^2)$ en el diseño de concepto $k$, de tal manera que $J_i(\boldsymbol{\theta}^1) \leq J_i(\boldsymbol{\theta}^2)$ para todos los $i \in [1, 2, \ldots, m]$ y todos los $k$ conceptos ($k \in [1, \ldots, K]$) y todos los $J_j(\boldsymbol{\theta}^1) \leq J_j(\boldsymbol{\theta}^2)$ para al menos un ($j, j \in [1, 2, \ldots, m]$) para cualquier concepto $k$.*

En consecuencia, el frente s-Pareto se construye con las alternativas de diseño de un conjunto $K$ de conceptos de diseño. En la Figura 4.5 se muestra esta noción. Se calculan dos aproximaciones diferentes del frente de Pareto para dos conceptos diferentes (distinguidos como círculos azules y cuadrados naranja respectivamente) como se puede ver en la Figura 4.5(a). En la Figura 4.5(b) se construye un frente s-Pareto con los dos conceptos de diseño.

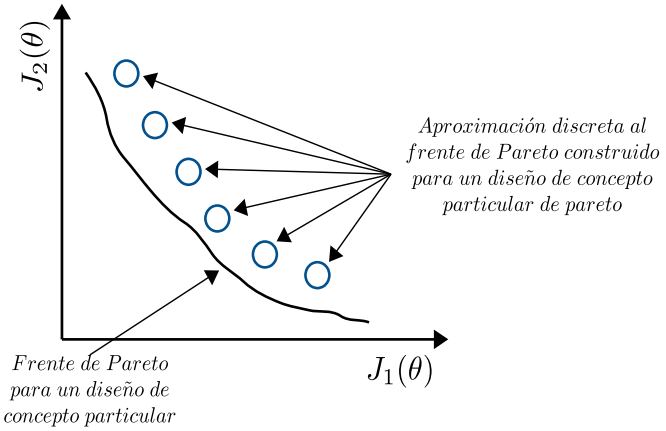
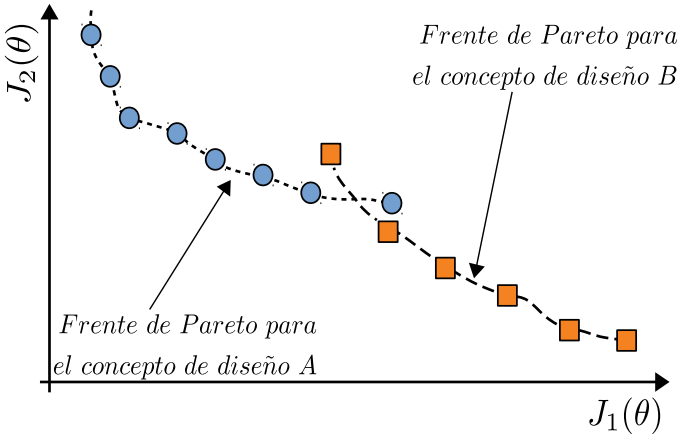


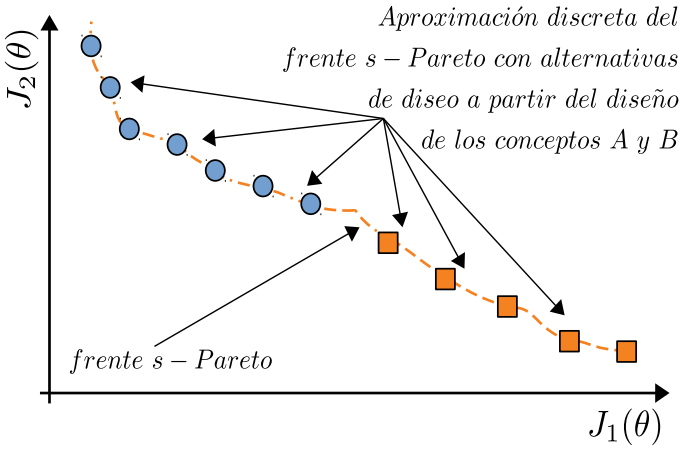

**Figura 4.4:** Diseño de Concepto y frente de Pareto para un diseño de concepto particular.

Por lo tanto, una comparación entre los conceptos de diseño es útil para el diseñador, porque así se podrán identificar los puntos fuertes, debilidades, limitaciones e inconvenientes de los conceptos [Mattson y Messac 2005]. Del mismo modo, es importante visualizar esas comparaciones y disponer de una medida cuantitativa para evaluar los puntos fuertes y débiles.

Finalmente, el concepto de diseño implica la existencia de un modelo paramétrico que define los valores de los parámetros (el espacio de decisión) que conducen a una determinada alternativa de diseño y su rendimiento (espacio objetivo). Lo que implica que la formulación del problema desde el punto de vista del diseñador no es la misma que desde el punto de vista del optimizador [Fonseca y Fleming 1998].




(**a**)Dos conceptos en el frente de Pareto.




(**b**)Frente s-Pareto, a partir de dos conceptos.

**Figura 4.5:** Definición del frente s-Pareto.

## 4.3 Conclusiones del capítulo

En este capítulo se han tratado algunos aspectos relativos a las definiciones del problema de optimización multiobjetivo MOP, completando de esta forma su descripción teórica y matemática.

Debido a que la generación de trayectorias en el espacio euclidiano es una tarea geométrica, entonces, para completar la resolución del problema de construcción de curvas suaves, continuas y navegables por un UAV (de acuerdo al capítulo 3), se ha propuesto un acercamiento multiobjetivo, detallado en el capítulo 9 [Samaniego, Sanchis, Garcia-Nieto y col. 2020], donde se determina un conjunto de los mejores puntos espaciales que garanticen la concatenación entre segmentos en forma de líneas rectas y segmentos de curvas (que parten de la forma de esferas), que además optimicen curvaturas y torsiones en cada uno de los segmentos y por lo tanto la curva total (después de unir segmentos). En relación con esto, la optimización multiobjetivo, devuelve resultados aceptables, dentro de los requerimientos establecidos, siempre que se realice una definición completa del MOP. Lo que significa que la propuesta de optimización desde un punto de vista geométrico 3D (sin desacoplar planos), hasta la integración de restricciones cinemáticas en la capacidad de vuelo de un UAV es alcanzable e incluso puede ser extendida, además que la correcta definición de la problemática devuelve resultados de aproximación de trayectorias aceptables en tiempos cortos.

Finalmente, cabe destacar que con una definición del MOP deficiente, las soluciones obtenidas no cumplirán las expectativas en la toma de decisiones, independientemente de la calidad de los algoritmos. Si el algoritmo es inadecuado para el problema en cuestión, el DM no obtendrá un conjunto de Pareto útil para analizar y, en consecuencia, no se podrá seleccionar una solución que satisfaga las necesidades requeridas.

# Capítulo 5

# Plataforma de test: Kadett 2400

***Resumen:*** *El presente capítulo inicia con una definición formal sobre el campo de los vehículos aéreos no tripulados UAV, también conocidos como Sistemas Aéreos No Tripulados, a continuación, se realiza una breve reseña histórica sobre esta rama tecnológica. Finalmente, se presentan las características de la plataforma de vuelo Kadett 2400 utilizada en este trabajo de tesis, así como su cinemática y dinámica.*

## 5.1 Introducción a los Vehículos Aéreos No Tripulados

Los Sistemas Aéreos No Tripuladas (UAS, del acrónimo en inglés *Unmanned Aerial Vehicle*) ó Vehículos Aéreos No Tripulados (UAV, del acrónimo en inglés *Unmanned Aerial Vehicle*), en la actualidad son un componente nuevo y evolutivo en el campo de la aviación que ofrece nuevas oportunidades y desafíos en la investigación científica. En este sentido, se hará la definición de la tecnología UAV, posteriormente, se presentará una breve reseña histórica sobre este campo.

**Definición 5.1.1** *Un **UAV** es un dispositivo destinado a ser utilizado para el vuelo, que no tiene un piloto a bordo. Este incluye todas las clases de aviones, helicópteros, aeronaves y aviones de elevación translacional que no tienen ningún piloto a bordo. Se entiende que las aeronaves no tripuladas incluyen*

*sólo las aeronaves controlables en tres ejes y, por lo tanto, excluyen los globos tradicionales. [Guidance 2008].*

**Definición 5.1.2** *Un **robot aéreo** es un sistema capaz de realizar un vuelo sostenido sin control humano directo y capaz de realizar una tarea específica [Siciliano y Khatib 2016].*

**Definición 5.1.3** *Un **UAV** es un vehículo propulsado que no lleva un operador humano, puede ser operado de manera autónoma o a distancia, puede ser prescindible o recuperable, y puede llevar una carga útil letal o no letal. Vehículos balísticos o semibalísticos, misiles de crucero, proyectiles de artillería, torpedos, minas, satélites, y sensores desatendidos (sin forma de propulsión) no se consideran vehículos no tripulados. Los vehículos no tripulados son el principal componente de los sistemas no tripulados. [Clapper y col. 2007].*

**Definición 5.1.4** *Un **Sistema Aéreo No Tripulado UAS** comprende elementos individuales del sistema que consisten en una "aeronave no tripulada", la "estación de control" y cualquier otro elemento del sistema necesario para permitir el vuelo. Puede haber múltiples estaciones de control, comando, enlaces de control y elementos de lanzamiento y recuperación de elementos dentro de un UAS [Clothier y col. 2011].*

De acuerdo con las anteriores definiciones, se puede describir un UAV como un dispositivo utilizado para tareas de vuelo sin obligatoriedad de piloto a bordo. No obstante, esta condición no excluye la existencia de un piloto remoto, controlador de misión u otra clase de operador.

## 5.2 Breve reseña histórica

A lo largo de las últimas dos décadas el desarrollo tecnológico de los UAVs ha sido muy elevado, partiendo en sus principios de equipos grandes, pesados y costosos, utilizados para la investigación académica, hasta llegar a ser unas herramientas desarrolladas para resolver una amplia gama de tareas en diferentes campos teóricos y prácticos, incluido entre ellos, los campos comerciales y militares. A pesar que su desarrollo ha sido vertiginoso, su origen se remonta más allá de las últimas tres décadas. No obstante, es importante resaltar que, en sus orígenes, estos sistemas no se podían calificar con la definición moderna de UAV. En este sentido, a continuación, se ofrece una breve perspectiva histórica de los vuelos no tripulados.

En el año 1915 la armada de los Estados Unidos desarrollo una primera aeronave no tripulada, esta fue un torpedo diseñado para volar y dirigirse a un objetivo específico. Durante la segunda guerra mundial, estas aeronaves fueron utilizadas con el objetivo de realizar misiones de reconocimiento. La Agencia de Proyectos de Investigación Avanzada de la Defensa DARPA y la NASA iniciaron investigaciones con el fin de determinar nuevos usos para los UAVs [Nonami y col. 2010; Angelov 2012], y de ese esfuerzo surgieron varios vehículos aéreos conocidos como Helios, Proteus, Altus Pathfinder y Predator, este último fue utilizado por primera vez por los EE.UU. en la Guerra del Golfo (estos vehículos aéreos se pueden apreciar en la Figura 5.1).

(**a**)UAV Helios desarrollado por la NASA y AeroVironment. Durante su segundo vuelo a gran altitud alcanzó los 96,863 pies. Fuente: Fotografía: NASA.

(**b**)Poteus, Avión experimental fabricado por Scaled Composites. Fuente: Jackson 2004.


(**c**)UAV Altus II de la NASA desarrollado por ERAST. Fuente: Fotografía NASA/General Atomics.


(**d**)El Ikhana es un UAV Predator B, adquirido por la NASA y adaptado para misiones científicas. Fuente: Fotografía de la NASA.

**Figura 5.1:** Vehículos aéreos no tripulados, resultantes de los proyectos de investigación de DARPA y la NASA.

Poco después del final de la Segunda Guerra Mundial, las misiones de reconocimiento fueron en aumento debido a su gran interés táctico. El primer avión teledirigido de reconocimiento fue el SD-1 [Newcome 2004] (ver Figura 5.2), también conocido como el MQM-57 Falconer, mismo que fue desarrollado a mediados de la década de 1950. El SD-1 era operado remotamente y llevaba una cámara integrada, y una autonomía de vuelo de 30 minutos [Zaloga 2011].

**Figura 5.2:** SD-1, también conocido como el MQM-57 Falconer, fue el primer dron de reconocimiento del Ejército de los EE.UU. y permaneció en servicio hasta los años 70. Fuente: National Museum of the USAF.

En la actualidad, los sistemas modernos (ver Figura 5.3) son mucho más diversos. Un ejemplo de evolución es el Pioneer, desarrollado en los años 80, o el SPERWER de construcción francesa, los cuales son sistemas más grandes, con mayores capacidades y de mayor resistencia como el RQ-4 Global Hawk. Del mismo modo se han desarrollado tecnologías que disponen de sistemas que pueden asumir múltiples funciones, como el MQ-9 Reaper, que además de reconocimiento puede ser utilizado como avión caza. Finalmente, se puede mencionar al Neptuno utilizado para operaciones acuáticas.

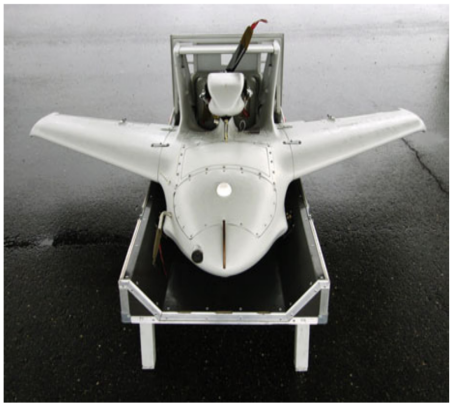
Por otra parte, los pequeños vehículos aéreos no tripulados también han suscitado un gran interés, puesto que han sido considerados como un punto de entrada hacia el mercado civil. Aunque su menor tamaño conlleva irremediablemente una reducción en la capacidad de carga útil, hay un gran número de UAVs pequeños y en miniatura en funcionamiento o en desarrollo activo. Esto se debe a que dicha tecnología es versátil, portátil y de fácil mantenimiento.

En este sentido, se debe resaltar que tienen capacidades de funcionamiento que pueden completar las mismas aplicaciones que los UAVs grandes, obviamente en menor escala y a un costo menor.

(**a**)Pioneer RQ-2 listo para ser lanzado durante la operación Escudo del Desierto. Fuente: Cuerpo de Marines US.


(**b**)El RQ-4 Global Hawk es un UAV de gran altitud y resistencia. Fue el primer UAV en hacer una travesía transpacífica. Fuente: Fuerza Aérea de los Estados Unidos, Sargento Mayor Jason Tudor.

(**c**)MQ-9 Reaper es una versión actualizada del UAV Predator. Se utiliza principalmente como un UAV persistente cazador-asesino. Fuente: Fotografía de US Fuerza Aérea, Sargento de Estado Mayor Brian Ferguson.



(**d**)Neptuno, es un UAV de reconocimiento capaz de realizar acuatizaje: Fuente: Fotografía Marina de los EE.UU.

**Figura 5.3:** Sistemas Modernos UAV.

De esta forma, se puede destacar la categoría de UAVs miniatura, existiendo sistemas con un peso entre $1 - 3kg$ y son fácilmente transportables (algunos incluso son plegables). Tales sistemas incluyen el Ciberbicho, Civeraves, etc.

Finalmente, es importante destacar que los sistemas mencionados anteriormente utilizan un diseño de ala fija. Sin embargo, existe un gran número de UAVs de tipo helicóptero, los cuales en la actualidad son utilizados en diferentes aplicaciones civiles y militares.

## 5.3 Breve Clasificación UAV

El constante desarrollo en el campo de la aviación no tripulada, ha dado lugar a diferentes terminologías, como Vehículo Aéreo No Tripulado UAV, o Sistema de Aviones No Tripulados UAS, terminologías que en el fondo describen una misma tecnología[1]. Por otra parte, en la guerra de Vietnam se utilizó el término RPVs o Vehículos Pilotados Remotamente. En la actualidad, la Fuerza Aérea de los Estados Unidos USAF ha sustituido la terminología RPV por la de Aviones pilotados a distancia o RPA (término utilizado para incluir tanto el avión como el piloto), mientras que el Reino Unido se los ha designado como Sistema Aéreo Pilotado a Distancia RPAS [Fishpool 2010; Herlik 2010].

De esta manera se puede considerar un elevado número de parámetros para realizar una clasificación de los UAVs, incluido el peso máximo al despegue MTOW, el tamaño, las condiciones de funcionamiento, las capacidades o cualquier combinación de estas y otras características. Cabe señalar que, si bien algunos de estos parámetros tienen un efecto mínimo en los requisitos de seguridad del sistema, siguen siendo importantes desde el punto de vista operacional, comercial, jurídico y, posiblemente, de otro tipo. A continuación, se presenta en la Tabla 5.1 una clasificación en base a las características mencionadas.

El MTOW, proporciona un buen sistema para clasificar las aeronaves, en este sentido, un esquema de clasificación simple que muestra el alcance previsto y la altitud máxima de funcionamiento de UAVs de diferentes clases se presenta en la Tabla 5.2.

Del mismo modo se puede hacer una clasificación basada en la altitud operativa y el riesgo de colisión en el aire (ver Tabla 5.3). De esta manera, los niveles de altitud han sido definidos en cuatro categorías, siendo:

[1] En la práctica, los términos UAS y los UAV se suelen utilizar indistintamente, no obstante, si el sistema es por demás importante (principalmente por razones legales/regulatorias) el término UAS tiene preferencia.

**Tabla 5.1:** Clasificación de UAVs sobre diferentes sistemas existentes. Fuente: [Blyenburgh 2006].

| | **Masa (kg)** | **Alcance (km)** | **Altura de vuelo (m)** | **Resistencia (h)** |
|---|---|---|---|---|
| Micro | $< 5$ | $< 10$ | 250 | 1 |
| Mini | $< 20/25/30/150^a$ | $< 10$ | 150/250/300 | $< 2$ |
| ***Tácticos*** | | | | |
| Poco alcance (CR) | $25 - 150$ | $10 - 30$ | $3,000$ | $2 - 4$ |
| Corto alcance (SR) | $50 - 250$ | $30 - 70$ | $3,000$ | $3 - 6$ |
| Medio alcance (MR) | $150 - 500$ | $70 - 200$ | $5,000$ | $6 - 10$ |
| Resistencia MR (MRE) | $500 - 1,500$ | $> 500$ | $8,000$ | $10 - 18$ |
| Baja altitud penetración profunda (LADP) | $250 - 2,500$ | $> 250$ | $50 - 9,000$ | $0{,}5 - 1$ |
| Baja altitud resistencia larga (LALE) | $15 - 25$ | $> 500$ | $3,000$ | $> 24$ |
| Altitud media larga resistencia (MALE) | $1,000 - 1,500$ | $> 500$ | $3,000$ | $24 - 48$ |
| ***Estratégicos*** | | | | |
| Gran altitud es larga resistencia (HALE) | $2,500 - 5,000$ | $> 2,000$ | $20,000$ | $24 - 48$ |
| Estratosférico (Strato) | $> 2,500$ | $> 2,000$ | $> 20,000$ | $> 48$ |
| Exoestratosfera (EXO) | TBD | TBD | $> 30,500$ | TBD |
| Combate no tripulado (UCAV) | $> 1,000$ | $1,500$ | $12,000$ | 2 |
| Letal (LET) | TBD | 300 | $4,000$ | $3 - 4$ |
| Señuelos (DEC) | $150 - 250$ | $0 - 500$ | $50 - 5,000$ | $< 4$ |

[a] Varía de acuerdo con las restricciones legales nacionales.
**TBD** $\rightarrow$ por determinarse, a convenir, a ser informado (o notificado), y por decidirse (del acrónimo en inglés *to be determined*).

1) Muy baja altitud (VLA/LOS), opera en el espacio aéreo de Clase G y típicamente en altitudes inferiores a $400 - 500$ pies, el operador mantiene contacto visual con la aeronave en todo momento.

2) Muy baja altitud (VLA/BLOS), posee las mismas características que la clase anterior, con la posibilidad adicional que la aeronave vuele más allá de la línea visual del operador.

3) Altitud media (MA), opera en Clase A, a través del espacio aéreo E.

4) Altitud muy alta (VHA), opera en el espacio aéreo de Clase E, por encima de $FL600$.

**Tabla 5.2:** Clasificación UAV basada en MTOW. Fuente: [Norbert Tränapp 2001].

| Clase | MTOW (kg) | Categoría de alcance | Máxima altitud (ft) |
|---|---|---|---|
| **0** | 25 | Poco alcance | $1,000ft$ |
| **1** | $25 - 500$ | Corto alcance | $15,000ft$ |
| **2** | $501 - 2,000$ | Medio alcance | $30,000ft$ |
| **3** | $> 2,000$ | Largo alcance | Sobre $30,000ft$ |

**Tabla 5.3:** Clasificación basada en la clase de espacio aéreo. Fuente: [Dalamagkidis 2015].

| Clase | Clase de espacio aéreo | S&A | Transpondedor | Comunicación ATC de 2 vías |
|---|---|---|---|---|
| VLA/LOS | Clase G | No requerido | No requerido | No requerido[a] |
| VLA/BLOS | Clase G | Requerido[b] | Requerido[b] | No requerido[a] |
| MA | Clase A-E | Requerido | Requerido | Requerido |
| MA/A | Clase A | Requerido[b] | Requerido | Requerido |
| VHA | Sobre Fl600 | Requerido[b] | Requerido | Requerido[b] |

[a] La comunicación con el ATC antes de la operación puede ser todavía necesaria.
[b] Se puede renunciar a ella para ciertos tipos de operaciones o en ciertas condiciones.

Otra interesante clasificación de los UAVs se basa en su nivel de autonomía. En la actualidad, los UAVs exhiben autonomía en determinadas funciones, y se prevé que esta tendencia aumente, tal como se analiza en [Protti y Barzan 2007]. En el año 2005 se propusieron los niveles de control autónomo ACL para medir autonomía [Clough 2002], estos diez niveles propuestos, se basaban en requisitos como el conocimiento de la situación, el análisis, la coordinación, la toma de decisiones y la capacidad operativa. Una lista ACL se presenta en la Tabla 5.4. Mientras que en la Tabla 5.5 se presenta una clasificación similar propuesta por [Protti y Barzan 2007], donde se menciona que los UAVs pueden mostrar un comportamiento no determinante.

1) Piloto a distancia: Un piloto certificado controla remotamente el sistema dentro de la LOS o con la retroalimentación de los sensores del UAV.

2) Operado remotamente (semi-autónomo): El UAV recibe comandos de alto nivel (waypoints, objetos a rastrear, etc.) y su desempeño es monitoreado por un operador entrenado. En este caso el vuelo es realizado por el propio UAV, pero toda la toma de decisiones son delegadas a un humano.

3) Completamente autónomo: El UAV recibe tareas generales y es capaz de determinar cómo llevarlas a cabo, incluso ante imprevistos. También puede vigilar su bienestar y tomar medidas correctivas sobre la aparición de fallos.

**Tabla 5.4:** Niveles de control Autónomo [Clough 2002].

| ACL | Descritpor de nivel |
|---|---|
| 0 | Vehículo pilotado a distancia |
| 1 | Ejecutar la misión planeada de antemano |
| 2 | Misión cambiante |
| 3 | Respuesta robusta a fallas/eventos en tiempo real |
| 4 | Vehículo de adaptación de fallos/eventos |
| 5 | Coordinación de múltiples vehículos en tiempo real |
| 6 | Cooperación multimodal en tiempo real |
| 7 | Saber del espacio de batalla |
| 8 | Conocimiento del espacio de batalla |
| 9 | Conocimiento de los enjambres de batalla |
| 10 | Completamente autónomo |

Finalmente, se puede realizar una clasificación adicional en base a su propiedad, como públicos o estatales (están operados por entidades públicas como organismos locales de aplicación de la ley) y civiles (son propiedad de la industria) o privados [Hempe 2006].

## 5.4 Kadett 2400

El *Graupner Kadett 2400 RC aircraft* (ver Figura 5.4) es un UAV fabricado por Graupner. Este UAV tiene una estructura muy ligera, sus características aerodinámicas lo hacen adecuado para los fines de esta investigación. Estas características incluyen una envergadura de 2.4 $m$, 0.9 $m^2$ de superficie alar, 48.07 $N/m^2$ de carga alar (este perfil de ala alta confiere un comportamiento en vuelo muy estable, característica que permite volar a bajas velocidades e incluso recuperar la estabilidad tras haber entrado en pérdidas), y 1.65 $\times 10^{-2} m^3$ de volumen disponible para el hardware de control.

Por otra parte, la Figura 5.5 ilustra la interconexión entre los dispositivos del UAV y el sistema de control de vuelo. La aeronave alberga todos los dispositivos necesarios para el control manual, así como automático. Durante el vuelo normal, el timón de cola, los elevadores y los alerones sirven como superficies

de control. La propulsión es proporcionada por un motor de corriente alterna sin escobillas alimentado por dos baterías de polímero de iones de litio LiPo a través de un variador de frecuencia. El variador y los servomotores están controlados por señales comando moduladas por Ancho de Pulso (PWM, del acrónimo en inglés, *Pulse Width Modulation*). El Controlador de Servo Interruptores (SSC, acrónimo del inglés *Servo System Controllers*) conmuta entre los modos de vuelo manual y autónomo lo que también permite la adquisición de datos y la aplicación de deflexiones de la superficie de control y cambios de par del motor.

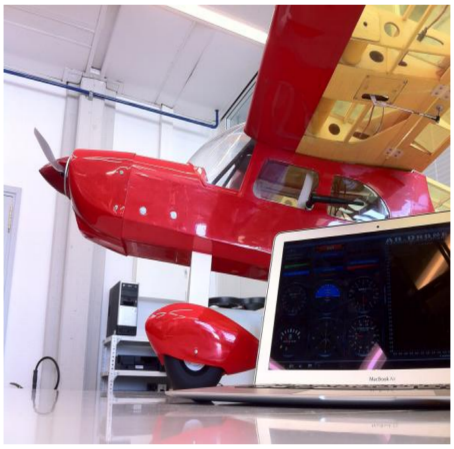
**Tabla 5.5:** Clasificación de UAV sobre sistemas existentes (Fuente: van Blyenburgh)

| **Clase** | **Categoría** | **Altitud de operación normal** | **Radio de misión normal** | **Plataformas de ejemplo** |
|---|---|---|---|---|
| Clase I (menos de $150kg$) | Small $> 20kg$ | Sobre $5,000ft$ AGL | $50km$ (LOS) | Luna, Hermes 90 |
| | Mini $2 - 20kg$ | Sobre $3,000ft$ AGL | $25km$ (LOS) | Scan Eagle, Skylark, Raven, DH3, Aladin, Strix |
| | Micro $< 2kg$ | Sobre $200ft$ AGL | $5km$ (LOS) | Black Widow |
| Clase II ($150 - 600kg$) | Táctico | Sobre $10,000ft$ AGL | $200km$ (LOS) | Sperwer, Iview 250, Hermes 450, Aerostar, Ranger |
| Clase III ($> 600kg$) | Strike combat | Sobre $65,000ft$ AGL | (BLOS) Ilimitado | |
| | HALE | Sobre $65,000ft$ AGL | (BLOS) Ilimitado | Global Hawk |
| | MALE | Sobre $45,000ft$ AGL | (BLOS) Ilimitado | Predator B, Predator A, Heron, Heron TP, Hermes 900 |

La Estación de Control de Vuelo (FCS, acrónimo del inglés *Flight Control System*), ha sido situada en un PC-104, misma que alberga los algoritmos de control. Una unidad IG500N de SBG Systems, integra una amplia gama de sensores, incluyendo acelerómetros, giróscopos y magnetómetros. Un filtro Kalman fusiona la información del sensor para estimar la posición, orientación, velocidad lineal y angular, y aceleración. La Tabla 5.6 proporciona los datos de precisión del fabricante para la unidad IG500N, plataforma que fue presentada

y probada en vuelo en [Velasco Carrau y col. 2012; Velasco Carrau 2014; Velasco y García-Nieto 2014].

(**a**)Plataforma de vuelo kadett 2400 (vista frontal).



(**b**)Plataforma de vuelo kadett 2400 (vista lateral).

**Figura 5.4:** Plataforma de vuelo Avión convencional *Kadett 2400* (Graupner). Elevadores, alerones y flaps de aterrizaje son accionados por servos instalados cerca de las superficies de control. Fuente: [Velasco Carrau y col. 2012]

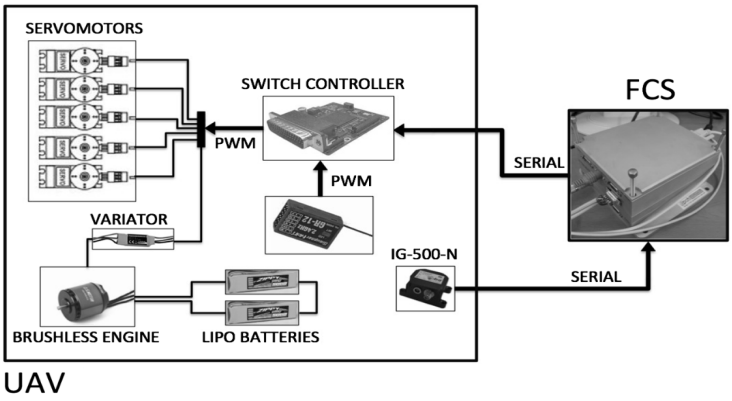



**Figura 5.5:** Dentro del cuadro a la izquierda se puede ver los enlaces de conexión y mecanismos de funcionamiento necesarios para controlar la dirección de vuelo del UAV, así como el controlador del motor y su velocidad. Mientras que a la derecha se puede ver el FCS PC-104. Fuente: [Velasco y García-Nieto 2014].

**Tabla 5.6:** Características de la unidad IG500N

| Sensor | Características | Valor |
|---|---|---|
| **Actitud de la unidad** | Precisión estática (Pitch) | ± 0.5 $deg$ |
| | Precisión estática (Roll) | ± 0.5 $deg$ |
| | Precisión estática (Heading) | ± 1.0 $deg$ |
| | Precisión dinámica | ± 1.0 $deg\ rms$ |
| **Acelerómetros** | No linealidad | <0.2 % de escala completa |
| | Estabilidad del sesgo | 5 $mg$ |
| **Giroscopios** | No linealidad | < 0.1 % de escala completa |
| | Estabilidad del sesgo | 0.5 $deg/s$ |
| **Magnetómetros** | No linealidad | <0.2 % de escala completa |
| | Estabilidad del sesgo | 0.5 $mG$ |
| **Receptor de GPS** | Precisión horizontal | 2.0 $m$ |
| | Precisión vertical | 5.0 $m$ |

### *5.4.1 Modelo dinámico de la aeronave*

Las ecuaciones del modelo dinámico de la aeronave están dadas por las ecuaciones de fuerza según se especifica en [H. Chang y col. 2018]:

$$\begin{aligned}
\dot{u} &= rv - qw + \frac{\overline{q}S}{m}C_X(\delta[e,a,r]) - g\sin\theta + \frac{T}{m} \\
\dot{v} &= pw - ru + \frac{\overline{q}S}{m}C_Y(\delta[e,a,r]) + g\cos\theta\sin\phi \\
\dot{w} &= qu - pv + \frac{\overline{q}S}{m}C_Z(\delta[e,a,r]) + g\cos\theta\cos\phi
\end{aligned} \tag{5.1}$$

donde $g$ es la intensidad del campo gravitacional cerca de la superficie de la Tierra, mientras que las ecuaciones de torque están definidas por,

$$\begin{aligned}
\dot{p} - \frac{I_{xz}}{I_x}\dot{r} &= \frac{\overline{q}Sb}{I_x}C_l(\delta[e,a,r]) - \frac{I_z - I_y}{I_x}qr + \frac{I_{xz}}{I_x}qp \\
\dot{q} &= \frac{\overline{q}S\overline{c}}{I_y}C_m(\delta[e,a,r]) - \frac{I_x - I_z}{I_y}pr - \frac{I_{xz}}{I_y}(p^2 - r^2) + I_p\Omega_p q \\
\dot{r} - \frac{I_{xz}}{I_z}\dot{p} &= \frac{\overline{q}Sb}{I_z}C_n(\delta[e,a,r]) - \frac{I_y - I_x}{I_z}pq + \frac{I_{xz}}{I_z}qr - I_p\Omega_p q
\end{aligned} \tag{5.2}$$

y las ecuaciones cinemáticas:

$$\begin{aligned}\dot{\phi} &= p + \tan\theta(q\sin\phi + r\cos\phi)\\ \dot{\theta} &= q\cos\phi - r\sin\phi\\ \dot{\psi} &= \frac{q\sin\phi + r\cos\phi}{\cos\theta}\end{aligned} \tag{5.3}$$

Para las ecuaciones 5.2 y 5.3, $m$ es la masa total del sistema. En la Figura 5.6, se puede ver el sistema de referencia cuerpo $(X_bY_bZ_b)$, $(u, v, w)$ son los componentes de la velocidad de traslación, $(p, q, r)$ son los componentes de velocidad angular, $(I_xI_yI_z)$ son los momentos de inercia mientras que $I_{xz}$ es el producto de inercia. Los productos de inercia $I_{xy}$ y $I_{yz}$ relacionados al plano longitudinal $(Y_b = 0)$ son nulos debido a la simetría del UAV con respecto a este plano. $I_p$ es la rotación inercial del motor y la hélice, $\Omega_p$ es la velocidad de rotación, y $T$ es el empuje del motor. $S$, $b$ y $\overline{c}$ son la superficie aerodinámica del modelo *Kadett 2400*, la envergadura y la cuerda alar respectivamente, y $\overline{q}$ es la presión dinámica[2]. Los coeficientes aerodinámicos (AC) $C_X, C_Y, C_Z, C_l, C_m$ y $C_n$ son funciones de las variables sistema. El símbolo $\delta$ representa la dependencia de las deflexiones de las superficies de control ($\delta_e, \delta_a$ y $\delta_r$ son las deflexiones de los elevadores, alerones y timón respectivamente). Finalmente, la orientación del UAV se representa por los ángulos de Euler alabeo $\theta$, cabeceo $\phi$ y guiñada $\psi$.

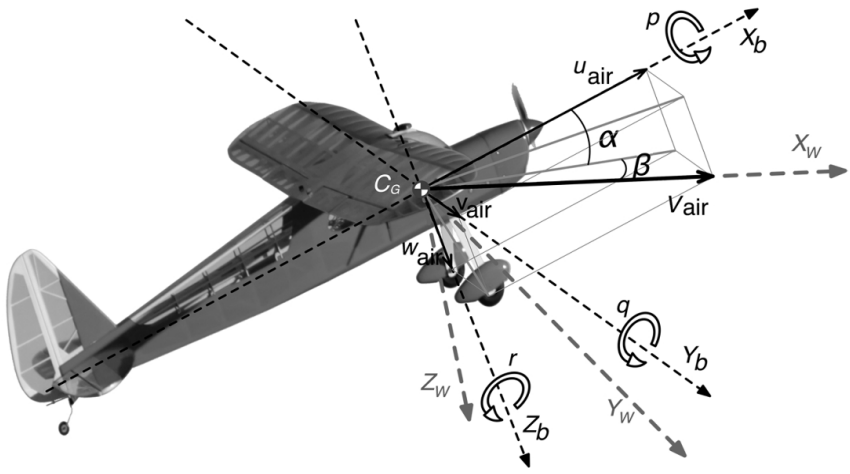



**Figura 5.6:** Los ejes de la carrocería de la aeronave, los ejes del viento de la aeronave y la velocidad relativa del viento. Fuente: [Velasco y García-Nieto 2014].

[2]Es una función de la la densidad del aire y la velocidad del aire en relación con el viento local

Si se asume un escenario en el cual el UAV se encuentra en condiciones de vuelo estable, como se detalla en [Klein y Morelli 2006], y no se realizan maniobras abruptas, entonces sus ecuaciones se mantienen en un primer y segundo orden. En tal sentido, si existen pequeñas perturbaciones, sobre la base de simetría del UAV se puede asumir que: 1) Las variables simétricas (longitudinales) $u$, $w$ y $q$ no afectan a las fuerzas y pares asimétricos (laterales) $Y$, $L$ y $N$; y de manera similar, 2) Las variables asimétricas (laterales) $v$, $p$ y $r$ no afectan a las fuerzas y pares simétricos (longitudinales) $X$, $Z$ y $M$. Finalmente, se presenta una aproximación de las ecuaciones aerodinámicas.

Modelos aerodinámicos longitudinales:

$$\begin{aligned} C_D(t) = {} & \mathbf{C_{D_0}} + \mathbf{C_{D_V}} \frac{1}{V_0} \Delta V(t) + \mathbf{C_{D_\alpha}} \Delta\alpha(t) \\ & + \mathbf{C_{D_{\alpha^2}}} \Delta\alpha(t)^2 + \mathbf{C_{D_q}} \frac{\overline{c}}{2V_0} q(t) + \mathbf{C_{D_{\delta_d}}} \Delta\delta_e(t) \end{aligned} \tag{5.4}$$

$$\begin{aligned} C_L(t) = {} & \mathbf{C_{L_0}} + \mathbf{C_{L_V}} \frac{1}{V_0} \Delta V(t) + \mathbf{C_{L_\alpha}} \Delta\alpha(t) \\ & + \mathbf{C_{L_{\alpha^2}}} \Delta\alpha(t)^2 + \mathbf{C_{L_{\dot\alpha}}} \frac{\overline{c}}{2V_0} \dot\alpha(t) \\ & + \mathbf{C_{L_q}} \frac{\overline{c}}{2V_0} q(t) + \mathbf{C_{L_{\delta_e}}} \Delta\delta_e(t) \end{aligned} \tag{5.5}$$

$$\begin{aligned} C_m(t) = {} & \mathbf{C_{m_0}} + \mathbf{C_{m_V}} \frac{1}{V_0} \Delta V(t) + \mathbf{C_{m_\alpha}} \Delta\alpha(t) \\ & + \mathbf{C_{m_{\alpha^2}}} \Delta\alpha(t)^2 + \mathbf{C_{m_{\dot\alpha}}} \frac{\overline{c}}{2V_0} \dot\alpha(t) \\ & + \mathbf{C_{m_q}} \frac{\overline{c}}{2V_0} q(t) + \mathbf{C_{q_{\delta_e}}} \Delta\delta_e(t) \end{aligned} \tag{5.6}$$

Modelos aerodinámicos laterales:

$$\begin{aligned} C_Y(t) = {} & \mathbf{C_{Y_0}} + \mathbf{C_{Y_\beta}} \Delta\beta(t) + \mathbf{C_{Y_p}} \frac{b}{2V_0} p(t) \\ & + \mathbf{C_{Y_r}} \frac{b}{2V_0} r(t) + \mathbf{C_{Y_{\alpha_{al}}}} \Delta\delta_{al}(t) + \mathbf{C_{Y_{\delta_r}}} \Delta\delta_r(t) \end{aligned} \tag{5.7}$$

$$\begin{aligned} C_l(t) = \mathbf{C_{l_0}} + \mathbf{C_{l_\beta}}\Delta\beta(t) + \mathbf{C_{l_p}}\frac{b}{2V_0}p(t) \\ + \mathbf{C_{l_r}}\frac{b}{2V_0}r(t) + \mathbf{C_{l_{\alpha_{al}}}}\Delta\delta_{al}(t) + \mathbf{C_{l_{\delta_r}}}\Delta\delta_r(t) \end{aligned} \tag{5.8}$$

$$\begin{aligned} C_n(t) = \mathbf{C_{n_0}} + \mathbf{C_{n_\beta}}\Delta\beta(t) + \mathbf{C_{n_p}}\frac{b}{2V_0}p(t) \\ + \mathbf{C_{n_r}}\frac{b}{2V_0}r(t) + \mathbf{C_{n_{\alpha_{al}}}}\Delta\delta_{al}(t) + \mathbf{C_{n_{\delta_r}}}\Delta\delta_r(t) \end{aligned} \tag{5.9}$$

donde $\alpha$ y $\beta$ son el ángulo de ataque y de desplazamiento lateral respectivamente, mientras $V$ es la velocidad del aire. En particular, $V_0$ es la velocidad del aire medida en estado de vuelo estable (antes que inicie una maniobra). Estas variables dependen de la velocidad y pueden ser calculadas de la siguiente manera:

$$\alpha = \arctan\left(\frac{w}{u}\right) \; ; \; \beta = \arcsin\left(\frac{v}{V}\right) \tag{5.10}$$

$$V = |\overrightarrow{V}| = \sqrt{u^2 + v^2 + w^2} \tag{5.11}$$

Además, $C_L$ y $C_D$ son los coeficientes de elevación y arrastre y su relación con $C_X$ y $C_Z$ se define como:

$$C_L(t) = -C_Z(t)\cos(\alpha(t)) + C_X(t)\sin(\alpha(t)) \tag{5.12}$$

$$C_D(t) = -C_X(t)\cos(\alpha(t)) + C_Z(t)\sin(\alpha(t)) \tag{5.13}$$

Entonces, el modelo aerodinámico del UAV está basado en las expresiones polinómicas de las constantes (marcadas en negrita) de las ecuaciones de vuelo por medio del modelo dinámico 5.4 a 5.9. Esas constantes se denominan constantes no dimensionales o derivadas de la estabilidad.

## 5.5 Conclusiones del capítulo

El primer objetivo de este capítulo es proporcionar una revisión del modelo *Kadett 2400* UAV utilizado en este trabajo de tesis, así como el material teórico necesario para permitir una representación matemática precisa del movimiento libre y controlado de un UAV de ala fija presentado como un modelo de cuerpo rígido. Los elementos fundamentales que se presentaron fueron, la cinemática,

la dinámica del movimiento y la definición de las fuerzas y momentos que actúan en el UAV. Además de las ecuaciones de 6DoF del movimiento que describen la cinemática y la dinámica del movimiento de un cuerpo rígido.

El capítulo proporciona una derivación detallada de las ecuaciones de movimiento usando el enfoque newtoniano. Asumiendo que un UAV de ala fija puede representarse como un cuerpo rígido que se mueve en un espacio inercial que permite la derivación de las ecuaciones de momento lineal y angular. De esta forma, una vez realizada la descripción matemática del UAV, el siguiente objetivo busca la realización de pruebas del sistema a partir del modelado de las ecuaciones descritas, a través del sistema de cómputo numérico y entorno de programación visual Matlab/Simulink, mismo que permite completar las correspondientes validaciones de diseño, además de la realización de pruebas de forma anticipada y frecuente. En relación a esto, en los capítulos 8 [Samaniego, Sanchís y col. 2018] y 9 [Samaniego, Sanchis, Garcia-Nieto y col. 2020], se han validado un conjunto de simulaciones de vuelo y seguimiento de trayectorias continuas y suaves (propuestas en los mismos capítulos 8 y 9), que además son realizables de acuerdo a las capacidades de maniobrabilidad en vuelo del modelo UAV descrito.

# Capítulo 6

# UAV Motion Planning and Obstacle Avoidance Based on Adaptive 3D Cell Decomposition: Continuous Space vs Discrete Space

**Nota sobre el capítulo:** El presente capítulo realiza un estudio del estado del arte en cuanto a las metodologías geométricas y la manera en que se divide el entorno de movimiento, haciendo un enfoque sobre el entorno 3D. El contenido de este capítulo aparece en el siguiente trabajo, como publicación de revista:

> Samaniego, F., Sanchis, J., García-Nieto, S., & Simarro, R. (2017, October). *UAV Motion Planning and Obstacle Avoidance Based on Adaptive 3D Cell Decomposition: Continuous Space vs Discrete Space*. In 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM) (pp. 1-6). IEEE.

***Abstract:*** *One important challenge in the current research of UAVs is the obstacle avoidance feature, a highly demanded capability in all kinds of UAV applications. Different algorithms that perform sampling tasks in continuous or discrete spaces are widely used in path planning. The Probabilistic Roadmap (PRM) and Rapidly Exploring Random Tree (RRT) variants have a stochastic essence, which allows them to obtain a response even in wide and complex environments. However, they present disadvantages related to computational cost and time convergence of results. In this paper a survey on sampling-based planners is presented, comparing algorithms with continuous and discrete sampling in 3D environments. On the other hand, two new variants are introduced: the Exact Cell Decomposition on Probabilistic Roadmap (ECD-PRM), as an extension of PRM method, and the Modified Adaptive Cell Decomposition (MACD) performing a reduced space decomposition. The comparison is performed attending to several criteria such as path cost, number of generated nodes and number of control points in the final path, whenever it has been reached.*

## 6.1 Introduction

Autonomous UAVs are characterized by their skills to accomplish any flight mission without human intervention. An increasing number of applications for this type of air vehicles, in both the military and civilian sectors are in continuous development, from terrestrial monitoring (Guoqing Zhou y Deyan Zang 2007), active surveillance, participation in humanitarian missions or natural disasters to firefighting (Maza y col. 2011).

One important challenge in the current research is the obstacle avoidance feature, a highly demanded capability in all kinds of UAV applications. This navigation task has been managed from different paradigmatic perspectives such as Planned-Arrival-Time (Potvin, Y. Xu y Benyahia 2006) or Time-and-Time-delay (Cheol Chang, Myung Jin Chung y Bum Hee Lee 1994).

Combination of translational robotics, path planning and obstacle avoidance is taking relevant importance within UAV applications. The main challenge in the development of flight planning algorithms is the incorporation of active obstacle avoidance strategies ensuring a continuous and smoother safe airworthiness

(in problems of holonomic and nonholonomic control) (Kolmanovsky, Ilya and McClamroch 1995). This challenge involves taking into account several aspects such as tangential accelerations, partial derivatives of their dimensional components, odometry, noise, etc.

Different methodologies based on the aforementioned literature have been proposed to complete the task of 3D navigation for UAVs: Ant Colony Optimization (ACO) (Cekmez, Ozsiginan y Sahingoz 2016), Genetic Algorithms (GAs) (Sahingoz 2014) or Voronoi diagrams (Pehlivanoglu 2012). In general, these works generate a bi-dimensional trajectory of displacement and do not take into account the vehicle's constraints.

This work presents a survey in sampling-based planners and proposes the calculation of airworthy trajectories in 3D spaces. The proposal includes the usage of Octrees (Nieuwenhuisen y Behnke 2014) in discrete 3D spaces applying a Modified Adaptive Cell Decomposition (MACD) algorithm. Obtained results have been compared other popular algorithms in the field of path planning.

This paper is split into the following sections: In section 6.2 a revision of path planning problem definition is complete, section 6.3 defines the sampling method based on continuous spaces, section 6.4 defines the sampling method based on discrete spaces, section 6.5 raises the operation about the local planner, section 6.6 shows simulation results and section 6.7 discusses the conclusions and future work.

## 6.2 Path planning problem definition

To define the problem of path planning and obstacles avoidance, a number of main concepts must be reviewed. A workspace defined as $W = \mathbb{R}^3$ can be split into a finite set of spaces occupied by different obstacles $C_{obs}$ and a set of different free spaces $C_{free}$ inside a set of $C_{spaces}$. The path planning task is defined as the ability to travel from an initial point $p_{Init}$ to a goal point $p_{Goal}$, passing through a path constructed by free spaces in $C_{free}$. In UAV context, the aircraft must be defined as a 3D solid object that can be translated and rotated in space and its configuration requires of six variables: $(x, y, z)$ for translations and $(\alpha, \beta, \sigma)$ for Euler angles.

Search space can be defined as a continuous or discrete space. Depending on this definition a type of path planning algorithm can be used. These methodologies have a computational complexity $NP$ (Nondeterministic Polynomial Time) (Agrawal, Kayal y Saxena 2004) and the data assignments to a con-

nectivity graph $G$ (Archdeacon 1996) are built along the algorithm iterations. The graph $G(V, E)$ is constructed by a non-empty set of vertices $V$ and edges $E$. Where $V$ is the set of collision-free vertices belonging to $C_{free}$ and $E$ is the set of neighboring pairs corresponding to $V$. The consecutive sequence of collision-free spaces $P$ resulting from the path planning algorithm, defines the trajectory that the UAV must follow. The construction of $G(V, E)$ goes through a series of inherent steps for every planner such as:

1 ***Sampling.*** The first step of a planner is the selection of samples. Random or quasy-random samples in a continuous space or shapes from a discrete space can be selected. Selected samples are continuously added to a tree or roadmap depending on the planner.

2 ***Metric.*** It is a mathematical measurement of distance in which the pairs of points are compared on a map to calculate the similarity of two vectors.

3 ***Neighbors Selection.*** In the search algorithm, the neighboring vertices are set to nearby vertices joined by one or more edges. In continuous spaces, the nearby vertices are determined based on the defined metric function. In discrete spaces, the neighborhood is determined in relation to the convex boundaries on the vertices belonging to the edges.

4 ***Collision Checking.*** It is a boolean function that returns state (true or false) if any intersection exists between occupied space and the object in motion.

Fig. 6.1 shows a typical sequence based on sampling motion planning reasoned on the concepts described above. A geometric space is presented in which collision checking is performed and a possible tracking trajectory is generated.

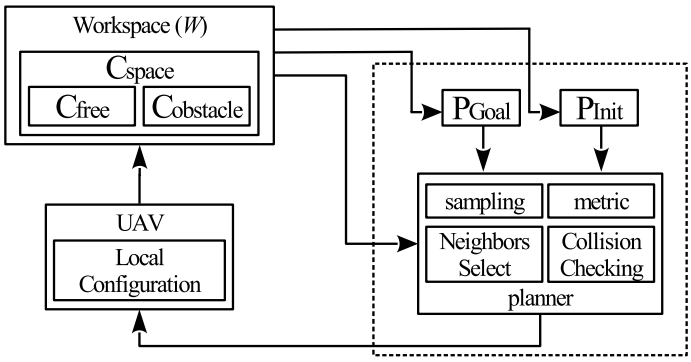



**Figura 6.1:** Planner based on sampling.

## 6.3 Randomness and continuous space sampling

Algorithms with stochastic nature try to guarantee a solution in complicated environments avoiding local minima. However, solutions will be generated only if they are reached within a predefined number of iterations.

Methods as Probabilistic Roadmap Method (PRM) (Amato e Y. Wu 1996) and Rapidly-exploring Random Trees (RRT) (Steven M LaValle 1998; LaValle, Steven M and Kuffner Jr 2000) have been applied successfully providing quality and intuitive path planning implementations in continuous spaces.

### *6.3.1 Probabilistic Roadmap Method (PRM)*

PRM belongs to sampling-based roadmaps methods. This planner executes a random sampling within a space navigation, where neighbors are connected according to a defined metric. The algorithm executes a number of iterations until the roadmap determines the connection between $p_{Init}$ and $p_{Goal}$ points. (see Algorithm 1).

**Algorithm 1** PRM Algorithm

**Require:** define params
**Require:** $G_i(V) \leftarrow p_{Init}$;
**Require:** $G_{i+1}(V) \leftarrow p_{Goal}$;
1: **while** i : N **do**
2: $p_{rand} = rand(x, y, z)$;
3: **if** $!(p_{rand}.collision)$ **then**
4: $G(V).add = p_{rand}$;
5: $planer.metric(G(V), p_{rand})$;
6: $G(E).add \leftarrow planner.connect(G(V), p_{rand})$;
7: **else**
8: $p_{rand}.discarded$;
9: **end if**
10: **end while**
11: **return** $G$;

In algorithm 1, a set of random points are generated inside the $C_{space}$ and added to the connectivity graph as long as they do not intersect any object in $C_{obj}$. Therefore, a collision-free point is added as a new vertex.

A connectivity graph is generated between neighborhood vertices according to the metric $d_E(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i,q_i)^2} \rightarrow P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_n)$. This is the first and more relevant phase in PRM, also known as the learning stage. At the end of iterations, the roadmap is generated.

### *6.3.2 Rapidly-exploring Random Trees (RRT)*

This algorithm builds an incremental tree based on random sampling. The algorithm begins with the generation of a random point with a distance bigger than the selected metric. After, the method locates the closest vertex of the incremental tree to previous generated random point and puts a new one between those points. (see Algorithm 2).

**Algorithm 2** RRT Algorithm

**Require:** define params
**Require:** $G_i(V).add \leftarrow p_{init}$;
**Require:** $G_{i+1}(V).add \leftarrow p_{Goal}$;
**while** $!(q_{new} = q_{goal})$ **do**
  $q_{rand} = rand(x, y, z)$;
  **if** $!(q_{rand}.collision)$ **then**
    $q_{near} \leftarrow findNearest.neighbors(G(V), q_{rand})$;
    $q_{new} \leftarrow planner.connect(G(V), q_{near})$;
    $G(V).add \leftarrow q_{new}$;
    $G(E).add \leftarrow planner.connect(q_{new}, q_{near}).edges$;
  **else**
    $q_{rand}.discarded$;
  **end if**
**end while**
**return** $G$;

The tree structure grows continuously until a number of predefined iterations is exceeded or until the connection procedure determines that the distance between $q_{new}$ and $q_{goal}$ is less than a predefined distance, making $(q_{new} = q_{goal})$.

## 6.4 Discrete space sampling

Similar to the digital signal sampling, where an analog signal measured in a range is sampled at a constant frequency for subsequent quantification, the $C_{spaces}$ is sub-divided into smaller spaces between its upper and lower limits.

Discrete space sampling in three dimensions builds a collection of nonoverlapping $k$ voxels (volume pixel) (Borgefors 1986; Borgefors, Hartmann y Tanimoto 1990) located in different $C_{spaces}$. Several techniques based on Octree have been applied for 3D reconstruction purposes (for example, making approximation and progressive compression with little loss of information on three-dimensional objects, either in colors or in grayscale (Samet y Kochut 2002)). However, this paper presents a different point of view, using Octrees for path planning and obstacles avoidance. In the voxel approach, the 3D world is represented by 3D boxes, indicating where the space is occupied and where it is not. Although its main drawbacks are the spatial dimension in the maximum decomposition of $C_{space} = \mathbb{R}^3$ and the potential generation of disconnected voxels with different internal properties, the voxels outer shape is standard and can be defined as a square, rectangle, pentagon, polygon, etc.

Further, voxel internal properties can be labeled and $C_{space}$ could be splitted in: $C_{free}$ whether the voxel is completely collision-free, $C_{full}$ whether the voxel is completely occupied by an obstacle and $C_{mix}$ whether the voxel is partially occupied.

### *6.4.1 Probabilistic Roadmap with Exact Cell Decomposition (PRM-ECD)*

In Exact Cells Decomposition (ECD) method, the set of voxels is defined taking into consideration the environment size. In a 3D approach, space is divided into voxels with identical outer shape and size properties. These series of voxels have got two main properties: $FREE$, whether the voxel is not occupied by an obstacle object i.e., $k_i \bigcap obj = 0$ and $NON - FREE$ whether the voxel space is partial or completely occupied by an obstacle object i.e., $k_i \bigcap obj = 1$.

The exact cell decomposition process determines the number of neighbors for each voxel (a number that oscillates between 3 and 6, considering neighbors whenever $k_i$ and $k_j$ shares boundaries). Since this process carries out an important computational cost, it has been obviated and the PRM learning stage has been adopted in ECD, generating random points with the dependence of the amount of voxels contained in the ECD workspace.

Random numbers do not refer to a 3D coordinate (x,y,z), but a voxel located in the workspace. Whether the randomly selected voxel does not collide with an obstacle, it is added to the connectivity graph. This process is repeated until the path that joins the $p_{Init}$ and $p_{Goal}$ points is generated. Fig 6.2. shows the beginning of the roadmap construction using Algorithm 3.

**Algorithm 3** ECD-PRM Algorithm

**Require:** $define\ params$;
**Require:** $T(V) \leftarrow environment.discrete$;
1: $G_i(V) \leftarrow T(V).locate(p_{Init})$;
2: $G_{i+1}(V) \leftarrow T(V).locate(p_{Goal})$;
3: **while** i : N **do**
4: $q_{rand} = rand(T(V))$;
5: **if** $!(q_{rand}.collision)$ **then**
6: $G(V).add = T(q_{rand})$;
7: $planner.metric(G(V), T(q_{rand}))$;
8: $G(E).add \leftarrow planner.connect(G(V), T(q_{rand}))$;
9: **else**
10: $q_{rand}.discarded$;
11: **end if**
12: **end while**
13: **return** $G$;

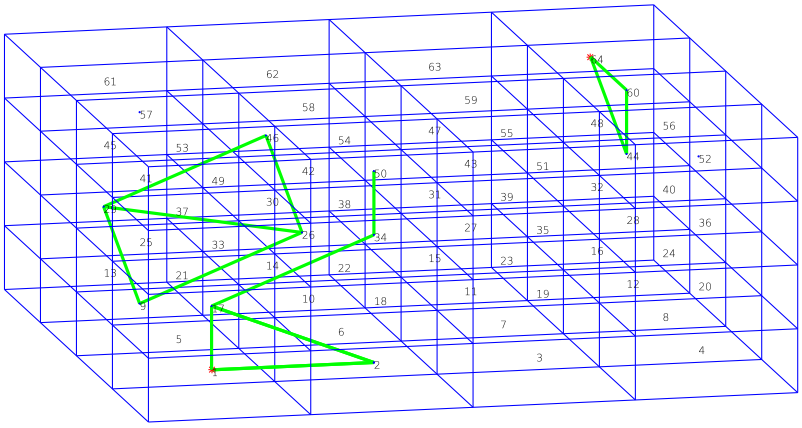


**Figura 6.2:** ECD-PRM. Building a roadmap. Learning stage.

The roadmap is built completely from the vertices of selected voxels during the random generation stage. This ensures that the UAV considers the central points of voxels free of collision and delimits the generation of random spaces in the environment.

### 6.4.2 Modified Adaptative Cell Decomposition (MACD)

ECD method generates an extra computational cost in the sampling process. The adaptive decomposition is more appropriate due to its numerical stability, complete processing assurance and its simple calculations (Pérez-rodríguez y Hernández-aguirre 2017). The decomposition is finished when the voxel size reaches a predefined fixed value.

The adaptive decomposition contains a finite set of joint or disjoint voxels $k_1, k_2, ..., k_n$ where $k_i$ and $k_j$ are adjacent whenever they share their boundaries. Voxels can be stated as: $FREE$ whether there is no occupied space around its boundaries (i.e. $k_i \bigcap obj = 0$), $FULL$ whether its limits are completely occupied by an object (i.e. $k_i \subseteq obj$) and $MIX$, if there is a partial occupation (i.e. $k_i \bigcap obj \wedge k_i \nsubseteq obj = 1$). Algorithm 4 shows this voxel nomination.

Notice that every voxel that has $C_{full}$ properties has not been stored. The goal is to find collision-free spaces $C_{free}$ or $C_{mix}$ spaces that can be sub-divided until the predefined fixed value $n$. Completing this hierarchical representation, more complex and compact, the number of neighbors for each voxel can be determined depending on its size, having a neighborhood of at least three voxels.

The connectivity graph $G$ contains information about the set of vertices (nodes) joined by edge links (arcs) allowing representation of a binary relation between the elements generated in the decomposition (Fig. 6.3).

**Algorithm 4** MACD Algorithm

**Require:** $n, env.zise(x, y, z)$;
**Require:** $obs = (x_{ObsInf}, x_{ObsUpp}, y_{ObsInf}, y_{ObsUpp}, z_{ObsInf}, z_{ObsUpp})$;
$vx = obs_{x_{Inf}} : env.size(x)/2^n : obs_{x_{Upp}}$;
$vy = obs_{y_{Inf}} : env.size(y)/2^n : obs_{y_{Upp}}$;
$vz = obs_{z_{Inf}} : env.size(z)/2^n : obs_{z_{Upp}}$;
$rectangloid.name = name$;
$rectangloid.boundary = env.size(x, y, z)$;
**for** $i = 1 : n$ **do**
  $rows = rectangloid.size$;
  **for** $nextRow : rows$ **do**
    **if** $rectangloid(nextRow).occup == mix$ **then**
      $boundary = boundaryOctree(rectangloid(nextRow).boundary)$;
      **for** $k = 1 : boundary$ **do**
        $rectangloid(newRow).name$;
        $rectangloid(newRow).boundary$;
        **if** $rectangloid(newRow).boundary(vx, vy, vz).obstacle == obstacle.mix$ **then**
          $rectangloid(newRow).occup == mix$;
        **end if**
        **if** $rectangloid(newRow).boundary(vx, vy, vz).obstacle == obstacle.free$ **then**
          $rectangloid(newRow).occup == free$;
        **end if**
      **end for**
    **end if**
  **end for**
**end for**

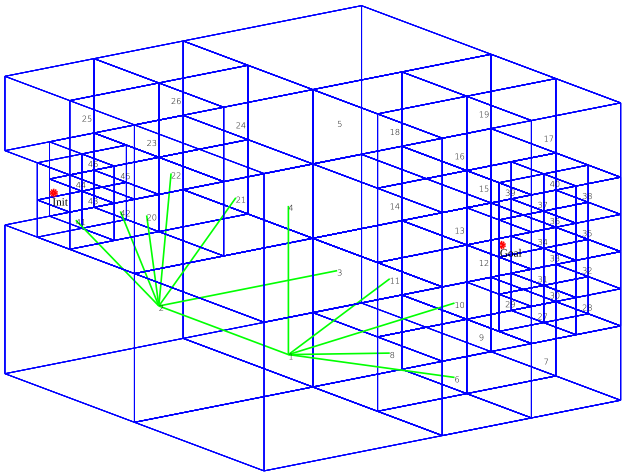


**Figura 6.3:** MACD and tree build since lowest node, decomposition level $2^n \leftarrow n = 3$.

The decomposition level shown in figure 6.3 is due to the relationship between the size of the environment and the number of levels to build. If standard decomposition methodology is determined in octants, then each sub-octant could contain eight new octants. Whether $2^n$ indicates the number of structure levels, the smallest voxel size is equal to $(x, y, z)/2^n$, being $(x, y, z)$ the total dimensions of the 3D environment.

## 6.5 Local Planner

Path planning defines a starting configuration $q_{init}$ and an arrival configuration $q_{Goal}$. The algorithm will generate a purely path with $C_{free}$ connecting $q_{init}$ and $q_{goal}$ comprising the following steps:

1 Locate the $C_{free}$ containers of $q_{init}$ and $q_{goal}$.

2 Perform the sampling algorithm until local conditions were satisfied.

3 Establish the shortest path between $q_{Init}$ and $q_{Goal}$ in $C_{free}$

The sampling process returns a wide range of $C_{free}$ spaces joined by edges. Path planning is completed in its initial phase by executing a search path algorithm that joins the $p_{Init}$ and $p_{Goal}$ points. Different path search parameters can be applied to complete this goal. For example, Dijkstra (Dijkstra 1959) and A* (Hart y Nils 1968) attempts an optimal solution in terms of distance through the construction of connectivity graphs. D* (Anthony Stentz 1995), AD* (Likhachev, Ferguson y col. 2008) and ARA* (Likhachev, Gordon y Thrun 2014) adopt a dynamic methodology of graph construction.

## 6.6 Experiments and Simulation Results

In this section, the results obtained by PRM, RRT, ECD-PRM and MACD algorithms described previously are presented and analyzed. All tests have been performed with Matlab (2015) under 64 bits Ubuntu Linux 16.04 LTS on a PC intel i7-4790 CPU @ 3.60GHz processor (Manufacturer: Gigabyte Technology Co., Ltd., Model: B85M-D3H) and 8 GB internal RAM.

The experiments attempts to determine which methodology is best adapted in a real 3D search path. Environments with and without obstacles have been defined within a 3D space with dimensions $(800, 800, 800)$.

In each experiment, the aircraft begins its flight at location $(x_1, y_1, z_1)$ and the algorithm will search a path until reaches its destination in another defined coordinate. An UAV with small dimensions SUAV (small UAV) and reduced maneuverability is considered. Furthermore, it is assumed that the space required by the SUAV to perform a right incidence between the pair of points is less than or equal to 50.

The stochastic nature of PRM, RRT and ECD-PRM methods forces to perform several simulations before a feasible response are obtained. Therefore, 6 different experiments have been performed with 10 executions each. MACD does require 10 executions because decomposition is based on the occupied space, so it generates the same results regardless the number of times the planner searches for a response in a defined scenario.

### *6.6.1 Experiments and discussions*

Two scenarios have been defined to carry out the experiments: a scenario without obstacles and the second one with an obstacle of dimensions $(200 \times 200 \times 200)$ located in the center of coordinates. Experiments 1, 2 and 3 have been executed on the obstacle-free scenario and experiments 4, 5 and 6 have been executed on the second scenario, (see Table 6.1).

**Tabla 6.1:** Characteristics of the scenarios.

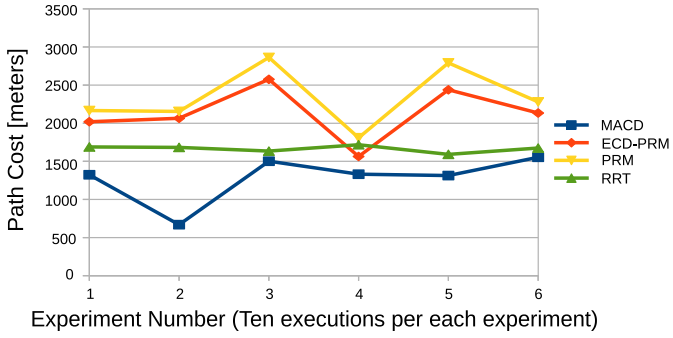
| | | | **Obstacle** | |
|---|---|---|---|---|
| **# Experiments** | $p_{Init}$ | $p_{Goal}$ | **yes** | **not** |
| **1** | (1, 1, 400) | (799, 799, 400) | | x |
| **2** | (100, 100, 200) | (799, 799, 600) | | x |
| **3** | (100, 100, 100) | (700, 700, 700) | | x |
| **4** | (1, 1, 400) | (799, 799, 400) | x | |
| **5** | (100, 100, 200) | (799, 799, 600) | x | |
| **6** | (100, 100, 100) | (700, 700, 700) | x | |

Notice that $p_{Init}$ and $p_{Goal}$ points have been set in different three axis coordinates forcing the planner to generate vertical and horizontal movements.

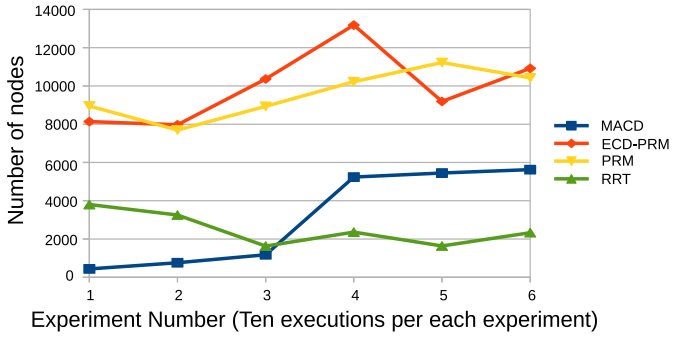
The construction of the roadmap in PRM has been done with vertex pairs that are located at distances less than or equal to 50. RRT has been configured as bidirectional and the $\epsilon$ value is dynamic (the distance generated is a value nearby 50) which determines the distance between $q_{near}$ and $q_{new}$. Before constructing

the roadmap for ECD-PRM, the exact cells decomposition has been executed with a value of $n = 6$. The minimum voxel size is $(800, 800, 800)/2^6$, sufficient to contain the SUAV. The roadmap will be constructed with distances close to 50. In MACD, where $n = 4$, the decomposition does not exceed the minimum size of 50, the voxel size minimum is $(800, 800, 800)/2^4$ and the number of level sub-voxels are equal to $2^4$.

Fig. 6.4(a) shows the Path Cost in meters $PC = \sum_{i=1}^{n} P_i$, resulting from average of executions performed by each planner. On one hand, PRM and ECD-PRM generate different cost values in their executions. On the other hand, RRT maintains a uniform cost, while MACD presents a lower cost. In Fig 6.4(b) the average of nodes created by each method can be appreciated. It can be seen how RRT and MACD generate fewer iterations than PRM and ECD-PRM.

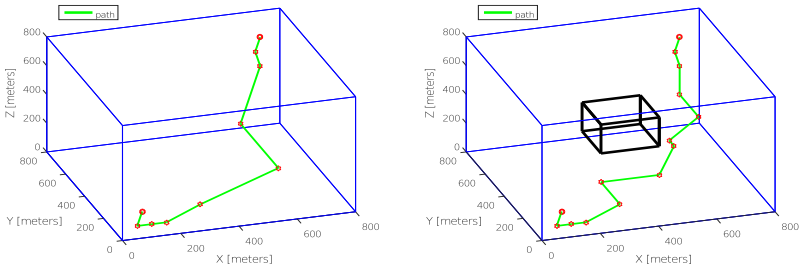



(**a**)Path cost over six executions.

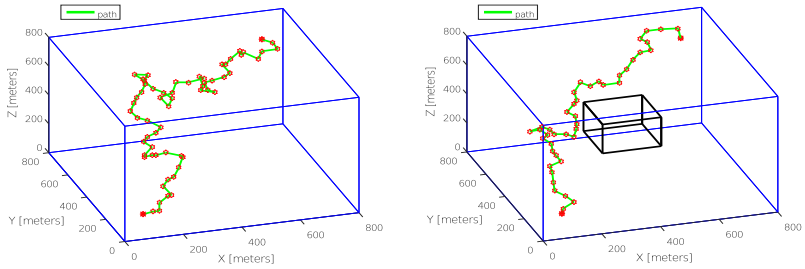



(**b**)Amount nodes generated over six executions.

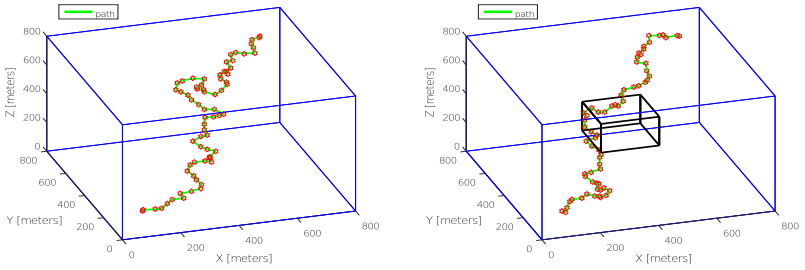
**Figura 6.4:** Comparison of different planners executions.



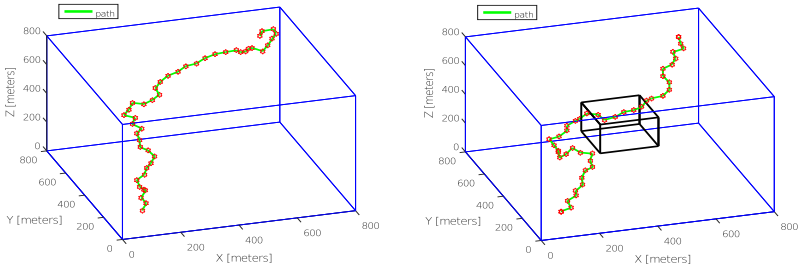

(**a**)MACD with and without obstacles.




(**b**)ECD-PRM with and without obstacles.




(**c**)PRM with and without obstacles.




(**d**)RRT with and without obstacles.

**Figura 6.5:** Path comparison between MACD, PRM and RRT algorithmics.

Fig. 6.5 depicts the resulting paths generated by PRM, RRT, ECD-PRM and MACD. The green line shows generated paths and red star line shows generated nodes. A 3D perspective $view(-18, 42)$ has been adopted in each scenario in order to appreciate differences. Fig. 6.5(a) shows the paths generated by MACD. Fig. 6.5(b) shows the paths consequence of ECD-PRM, Fig. 6.5(c) shows the PRM paths results and Fig. 6.5(d) shows the RRT paths produced.

From figures it can be seen that the set of nodes resulting from path planning is higher in PRM, RRT and ECD-PRM than in MACD. In addition, the randomness in PRM, RRT and ECD-PRM creates errant trajectories. MACD generates a smaller set of nodes besides a direct path between target and points. Further, for the second scenario, MACD tends to approach the obstacles, since a metric based on distance is used and voxels with smaller size optimize the search.

Fig. 6.6 highlights the number of nodes resulting from the final search process of each planner. Since MACD uses adaptive decomposition, there are free-space with large dimensional characteristics, causing a decrease of the set control nodes.

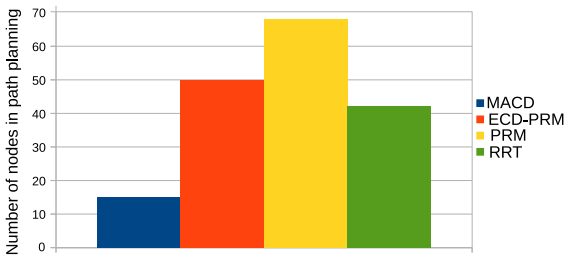



**Figura 6.6:** Number of Nodes (Control Points).

## 6.7 Conclusions and Future Work

Methods based on stochastic operators are widely used, and shown achievable results. However, the same stochastic core can result in a non-convergent response. The result may be an unfeasible path, reason why a mandatory post-processing is required. These methods generate a high amount of nodes and computational effort. In contrast, adaptive decomposition generates collision-free spaces with different dimensional characteristics, each voxel decreases or increases based on proximity to an obstacle, so if the UAV is approaching to

an obstacle, the voxel is reduced and if it moves away from the obstacle the voxel grows. This feature produces a low set of nodes.

In this paper a methodology for UAV motion and obstacle avoidance is presented. It reduces the computational effort in the generation of computational structures, reducing the resulting vertices from the final search path in order to increase nonholonomic constraints and degrees of freedom for the UAV movement. Future work in development of algorithms for UAV guidance and navigation, exploiting these nodes as control points for a future smooth curve approximation, will be done.

## Capítulo 7

# Recursive Rewarding Modified Adaptive Cell Decomposition (RR-MACD): A Dynamic Path Planning Algorithm for UAVs

**Nota sobre el capítulo:** El presente capítulo plantea la resolución al problema de búsqueda de planificación de trayectoria 3D al dividir el entorno 3D, ubicando el UAV en un espacio libre y asignando posibles recompensas a los espacio libres de colisión adyacentes al UAV. El espacio libre de mejor recompensa será al cual el UAV se desplace, posterior a ello el proceso se repetirá hasta alcanzar el punto objetivo. El contenido de este capítulo aparece en el siguiente trabajo:

> Samaniego, F., Sanchis, J., García-Nieto, S., & Simarro, R. (2019). *Recursive Rewarding Modified Adaptive Cell Decomposition (RR-MACD): A Dynamic Path Planning Algorithm for UAVs*. Electronics, 8(3), 306. DOI:https://doi.org/10.3390/electronics8030306.

***Abstract:*** *A relevant task in unmanned aerial vehicles (UAV) flight is path planning in 3D environments. This task must be completed using the least possible computing time. The aim of this article is to combine methodologies to optimise the task in time and offer a complete 3D trajectory. The flight environment will be considered as a 3D adaptive discrete mesh, where grids are created with minimal refinement in the search for collision-free spaces. The proposed path planning algorithm for UAV saves computational time and memory resources compared with classical techniques. With the construction of the discrete meshing, a cost response methodology is applied as a discrete deterministic finite automaton (DDFA). A set of optimal partial responses, calculated recursively, indicates the collision-free spaces in the final path for the UAV flight.*

## 7.1 Introduction

The world market for unmanned aerial vehicles (UAVs) is expanding rapidly, and there are various forecasts and projections regarding the market for unmanned vehicles. The economic impact of integrating UAVs into the National Airspace System in the United States will grow substantially and reach more than $82,1 billion between 2015 and 2025 (Valavanis y Vachtsevanos 2015).

A wide diversity of air missions can be completed by UAVs (*20 great UAV applications areas for Drones* 2014; *UNMANNED APPLIED SOLUTIONS* 2018) in various scenarios and including outdoor/indoor and water/ground/air/space environments (Rodríguez y col. 2013). The types of missions include military (missile launching drones, bomb-dropping drones, flying camouflaged drones) and civilian (video-graph/photography, disaster response, environment and climate) (J. Li y Han 2017; Stuchlík y col. 2015; Pulver y R. Wei 2018; Claesson y col. 2017; Reineman y col. 2013). A highly demanded task for UAVs is $3D$ autonomous navigation (either in static or dynamic environments) that optimises the route and minimises the computational cost. Thus, path planning defines the methodology that an autonomous robot must complete to move from an initial location to a final location, deploying its own resources as sensors, actuators, and strategies, while avoiding obstacles during the trip. Several path planning and obstacle avoidance techniques are being used in unmanned ground vehicles (UGVs), autonomous underwater vehicles (AUVs), and unmanned aerial vehicles (UAVs).

From the traditional robotics point of view, numerous works have been developed in which the path planning and obstacle avoidance algorithms perform searches in continuous or discrete Euclidean (Hernández, Zurro y Vazquez 2012) dimensional movement environments. It is important to mention that *LaValle* in (Lavalle 2006) has done significant work on sampling-based path planning algorithms. However, although his analysis is complete from a two-dimensional perspective, $3D$ planning analysis is not completely addressed. An exhaustive study of the growing work on sampling-based algorithms is presented in (Elbanhawi y Simic 2014). It must be remembered that the $2D$ path planning problem is NP-hard; and so environmental dimensional increases and UAV kinematics affect problem complexity.

An in-depth review of the current literature shows several works focus on two-dimensional ($2D$) scenarios (Hernandez, Bacca y Posso 2017) that limit vehicle behaviour to just a flat surface and consider its height as constant by making a dimensional analysis ($2{,}5D$) (Kohlbrecher y col. 2011). However, in complex unstructured situations (including, for example, forests, urban, or underwater environments) a simple $2D$ algorithm is insufficient and $3D$ path planning is needed.

A diversity of methodological paradigms have been developed to complete the task of $3D$ path planning. These are based on sampling, node/based algorithms, bio-inspired algorithms, and mathematical models, among other techniques. A brief bibliographic review focused on $3D$ trajectory planning is presented below.

Some representative techniques used in path planning methods and based on continuous and discrete environment sampling include: RRT (rapidly-exploring random tree) (Abbadi y col. 2012a; W. Aguilar y S. Morales 2016; W. G. Aguilar y col. 2017; Yao, Honglun Wang y Su 2015); PRM (probabilistic road maps) (Yan, Y.-S. Liu y Xiao 2013; Yeh y col. 2012; Denny y Nancy M. Amatoo 2013b; Q. Li y col. 2014b; Ortiz-Arroyo 2015a); Voronoi diagrams (Thanou y Tzes 2014; Y. Qu, Yintao Zhang y Youmin Zhang 2014; Fang, Luan y Z. Sun 2017); and artificial potential (Khuswendi, Hindersah y Adiprawita 2011; Xia Chen y J. Zhang 2013; Rivera, Prieto y Ramirez 2012; Liu Lifen y col. 2016). Nevertheless, it is important to note that RRT and PRM make random explorations (continuous sampling) of the defined environment. RRT is an expensive algorithm in terms of computational cost when searching for feasible solutions in cluttered environments. It should be emphasised that once the PRM road map is made, a methodological base built on nodes must be invoked to define the lowest cost path. The main disadvantage of the Voronoi diagram is

that it is an offline method. Finally, artificial potential algorithms present little computational complexity—although they tend to fall into local minimums.

Node-based algorithms (discrete space) are mathematical structures used to model pairwise relations (in this context, the structures are made with vertices and edges) and the aim is to calculate the cost of exploring nodes to find the optimal path. Various methodologies and subsequent variations, such as Dijkstra's algorithm (Dijkstra 1959; Verscheure y col. 2010), A* (Hart y Nils 1968; Niu y Zhuo 2008), D* (Anthony Stentz 1994; Ferguson y Anthony Stentz s.f.), and Theta* (De Filippis, Guglieri y Quagliotti 2012), present these characteristics in their results. In (Nosrati, Karimi Hojat Allah Hasanvand y Original 2012) the characteristics and approximations of various methodologies of * (Star) search algorithms are studied.

In recent years, these classical techniques have been improved with new learning machine techniques. ANN (Artificial Neural Networks) (Kroumov y Jianli Yu 2009; Gautam y Verma 2014; Maturana y Scherer 2015), fuzzy logic (Iswanto, Wahyunggoro e Imam Cahyadi 2016; S. Liu, Y. Wei y Gao 2012), ACO (Ant Colony Optimisation) (Duan, Yu y col. 2010; He y col. 2013), and PSO (Particle Swarm Optimisation) (Yudong Zhang, L. Wu y S. Wang 2013; Goel y col. 2018), among others (YongBo y col. 2017; G.-G. Wang, Chu y Mirjalili 2016; Aghababa 2012), are examples of these heuristic methodologies. Hence, these biological algorithms attempt to optimise the path by mimicking animal behaviour. The weaknesses and strengths of a set of heuristic techniques are discussed in (Mac y col. 2016). The implementation relevance of these methodologies does not present significant experimental results. In addition, the different techniques presented in this section have a particular computational cost and complexity based on the different approaches (Szirmay-Kalos y Márton 1998) (see Table 7.1).

**Tabla 7.1:** Computational cost and complexity in the graph structure, where $n$ is the number of vertex and $m$ is the number of edges.

| Method | Time Complexity | Memory | Real Time |
|---|---|---|---|
| Sampling based algorithms | $O(n\ log\ n)$ | $O(n^2)$ | On-line |
| Node based algorithms | $O(m\ log\ n)$ | $O(n^2)$ | On-line |
| Bioinspired algoritms | $O(n\ log\ n)$ | $O(n^2)$ | Off-line |

A summary of the above mentioned methodologies is shown in Table 7.2, which details the approximation methodology, authors and reference, type of obstacle

avoidance (static or dynamic), type of implementation (simulation or real), and publication year.

**Tabla 7.2:** 3D path planning methodologies studied list.

| Approach | Authors | Static Obstacle | Dynamic Obstacle | Simulation | Real | Year |
|---|---|---|---|---|---|---|
| RRT | Abbadi y col. 2012a | x | x | x | o | [2012] |
| | W. Aguilar y S. Morales 2016 | o | x | x | x | [2016] |
| | W. G. Aguilar y col. 2017 | x | o | x | x | [2017] |
| | Yao, Honglun Wang y Su 2015 | x | o | x | o | [2017] |
| PRM | Yan, F. Yan, Y.-S. Liu y Xiao 2013 | x | o | o | x | [2013] |
| | Yeh y col. 2012 | x | o | x | o | [2012] |
| | Denny y Nancy M. Amatoo 2013b | x | o | x | o | [2013] |
| | Q. Li y col. 2014b | x | o | x | o | [2014] |
| | Ortiz-Arroyo 2015a | x | o | x | o | [2015] |
| Voronoi | Thanou y Tzes 2014 | x | o | x | o | [2014] |
| | Y. Qu, Yintao Zhang y Youmin Zhang 2014 | x | o | x | o | [2014] |
| | Fang, Luan y Z. Sun 2017 | x | o | x | x | [2017] |
| Artificial Potencial | Khuswendi, Hindersah y Adiprawita 2011 | x | x | x | o | [2011] |
| | Xia Chen y J. Zhang 2013 | x | x | x | o | [2013] |
| | Rivera, Prieto y Ramirez 2012 | x | x | x | o | [2012] |
| | Liu Lifen y col. 2016 | x | x | x | o | [2016] |
| ANN | Kroumov y Jianli Yu 2009 | x | o | x | o | [2010] |
| | Gautam y Verma 2014 | x | o | x | o | [2014] |
| | Maturana y Scherer 2015 | x | o | o | x | [2015] |
| Fuzzy Logic | Iswanto, Wahyunggoro e Imam Cahyadi 2016 | x | x | x | o | [2016] |
| | S. Liu, Y. Wei y Gao 2012 | x | o | x | o | [2012] |
| ACO | Duan, H. Duan, Yu y col. 2010 | x | o | x | o | [2010] |
| | He, Y. He y col. 2013 | x | o | x | o | [2013] |
| PSO | Yudong Zhang, L. Wu y S. Wang 2013 | x | x | x | o | [2013] |
| | Goel y col. 2018 | x | x | x | o | [2018] |
| Others | YongBo y col. 2017 | x | o | x | o | [2017] |
| | G.-G. Wang, Chu y Mirjalili 2016 | x | o | x | o | [2016] |
| | Aghababa 2012 | x | o | x | o | [2012] |

The $3D$ path planning problem is still an open issue in this field. The general approach is to combine several of the above mentioned techniques to improve overall performance.

Several planners optimise the path planning distance. However, this paper attempts to include distance as an objective, as well as the geometrical characteristics of the UAV and flight constraints (velocity, turning capacity, battery, flight distance, etc.). All of these constraints are evaluated as potential cost and the path planning result is based on the sum of contributions for each cost (see Section 7.4). It is important to highlight that planning results do not attempt to arrive at an optimal path in a shorter distance. Furthermore, unlike other path planning methodologies in which pruning of the results is necessary, this paper attempts to minimise such pruning.

An adaptive cell decomposition (ACD) is a strong methodology for solving physical systems led by partial differential equations (Berger y Oliger 1984; Min y Gibou 2006). Such techniques offer a substantial improvement in computational time and discretisation is not governed by a dominant equations system. This methodology is used in accurate complex $3D$ Cartesian geometry reconstructions (Hasbestan y Senocak 2018; Pantano y col. 2007). The approach presented in this work does not seek a refined environment reconstruction, and only tries to determine occupied and free spaces within the $3D$ Cartesian space. The savings in computational and memory effort is significant. The computational structure that constructs the algorithm makes a rapid labelling of the geometric figure of the environment as a $3D$ solid with a rectangular shape.

In this paper, a functional $3D$ UAV path planning algorithm is proposed that is based on an evolution of the (ACD) method. The proposal attempts to achieve a linear speedup, exploring and decomposing the $3D$ environment under a recursive reward cost paradigm, and building an efficient and simple $3D$ path detection. The aim is not to generate a large scale reconstruction of the environment, nor start the procedure with a defined cloud of points (Ryde y Hu 2010). In the presented paper, the UAV just receives the obstacle information from the control station and generates a trajectory. Over time, physical phenomena often generate unknown space distributions between the UAV and obstacles, and so adaptation according these changes and spatial constraints might exist. In the event of obstacle collision with the previously calculated path, a new estimated path is generated.

This paper is organised as follows. Section 7.2 defines the terrain representation and codification obstacles, and the general problem of path planning under static and dynamic environments is stated. In Section 7.3, the basis of the adaptive cell decomposition technique is revisited and modified to be ready for our proposal. In Section 7.4, the new algorithm for planned paths is then explained, and finally, several application examples are shown in Section 7.5. Conclusions and future works are considered in the final section.

## 7.2 Problem Definition

At the moment when obstacles in the $3D$ real world are represented, they do not possess exact geometries, and for simplicity in this paper, obstacles have been modelled as cuboids in direct relation to their dimensional characteristics and location. When cuboids increase or decrease, special care is taken to ensure that they do not collide with each other and there is a free flight route during the environment tests.

In a $3D$ environment where a UAV performs a continuous flight from an *init* point ($q_i$) to a *goal* point ($q_f$), a set of various manoeuvres to complete this mission are deployed. The UAV had previously defined the trajectory to follow after taking into account several considerations and constraints.

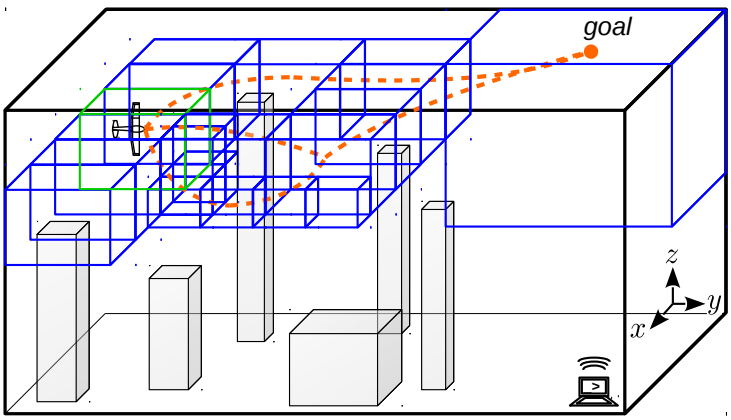



**Figura 7.1:** General scenario for $3D$ unmanned aerial vehicles (UAV) path planning. The grey cubes depict the environmental obstacles. Blue cubes are generated by the environmental discretisation and represent collision-free spaces. Orange lines are possible paths.

Let us assume that a complete description of the possible operating environment as an urban space in which buildings of different dimensions are defined. The UAV in flight receives data from its control station about the environmental conditions and it makes the necessary calculations to determine the best trajectory. The relevant data includes the goal point $q_f$, the current location, and the size of the obstacles (static or dynamic), as well as speed and movement directions. Since $q_i$ is related directly with the current UAV location, the aim is to apply a discrete decomposition (partial and recursive) of the environment to find the set of collision-free spaces for the UAV flight and head

towards the middle of those collision-free spaces until it arrives at $q_f$. Hence, the final trajectory result of this methodology generates a vector $(x_i, y_i, z_i)$ of three-dimensional points (system coordinates) translated as waypoints. Furthermore, it is important to emphasise that the resulting vector indicates spatial positions, and therefore the UAV that performs the trajectory tracking must possess these tracking skills. Thus, a UAV that has a quadropter-type holonomic system (the number of controllable degrees of freedom of the UAV system is equal to the total degrees of freedom) can complete the $3D$ waypoint tracking. Hence, a non-holonomic (the system is described by a set of parameters subject to differential constraints) (fixed wing) UAV does not have to be able to follow these trajectories.

Figure 7.1 shows an environmental example, where the discrete decomposition is built around the obstacles that interfere with the UAV flight. Hence, the three-dimensional characteristics of the obstacle might be considered to determine an escape trajectory that can surround the obstacle (including its sides and above and below) in continuous flight. Therefore, the $3D$ environment decomposition will take advantage of the $3D$ displacement capabilities of the UAV.

## 7.3 Modified Adaptive Cell Decomposition (*MACD*)

A standard $3D$ ACD algorithm attempts to make a discrete approximation of the environment, typically in a tree data structure known as octree (Octree is a tree data structure in which each internal node has exactly eight children) (Hanan Samet y Andrzej Kochut 2002). This process requires considerable computational resources and time. The modified adaptive cell decomposition *(MACD)* (Samaniego, Sanchis, Garcia-Nieto y col. 2017) does not make an exhaustive routing for each little space in the environment. If an obstacle exists, the routing in the three dimensions is in direct relation to the obstacle characteristics. The parameterised decomposition level is fixed in direct relation to the UAV manoeuvrability. This means that whenever the UAV needs a minimal space to complete a movement, this value will be defined in the level of decomposition.

Let us say $\Upsilon = (x, y, z)$ denotes a discrete $3D$ environment, where the set of collision-free voxels are defined as $S_{free}$, the set of occupied voxels are defined as $S_{occup}$, and the optimal path (metric term of distance) between $q_i$ and $q_f$ as $\rho$. The aim is to find the optimal path $\rho$ compound with a set of the nearest voxels in the $S_{free}$ space that enclose the obstacles.

The procedure is summarised in the flowchart shown in Figure 7.2, beginning with a specification of the number of decomposition levels. For the first level ($i = 0$), search limits are set to the environmental dimensions $\Upsilon$. Partition of the octree divides the environment in 8 equal parts in relation to the previous boundaries. In every single octree, an obstacle search is performed that determines the possibility of a later decomposition. Once this level decomposition is finished, the complete information of $S_{occup}$ and $S_{free}$ space is completed. Finally, a planner determines the best trajectory in distance terms.

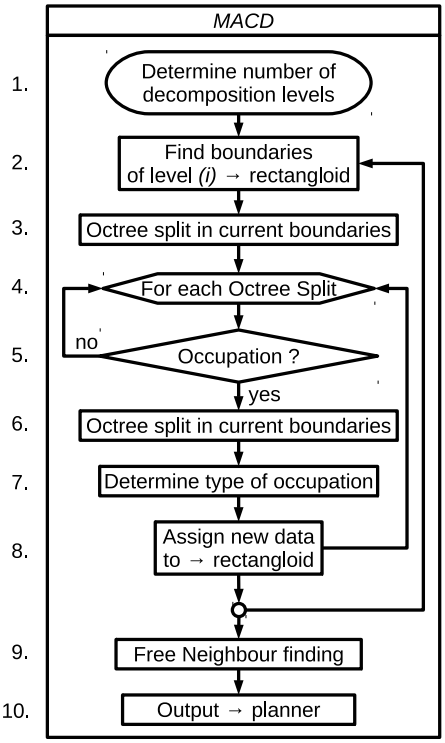



**Figura 7.2:** Flowchart of modified adaptive cell decomposition (*MACD*).

A simple example in a $2D$ environment is shown in Figure 7.3, using *quadtree* decomposition. The decomposition level $n$ is predefined, $n$ being the total levels which define the tree growth as a hierarchical computational structure. In a first segmentation, with $n = 1$, the environment is partitioned in $(2^n)^2$ underlying discrete spaces (blue lines). A second division of the environment ($n = 2$) will generate 16 spaces (green lines).

Figure 7.3 shows an example of a *quadtree* decomposition, denoted by $q_k$ with current level $n = 0$, with reference to its own neighbourhood. On its left side, there are neighbours with smaller dimensional characteristics than the present $q_k$. There is a total of 8 neighbours, from $(2^n)^2$ with $n = 3 \rightarrow (2^3)^2 = 64$, of which only 8 are related directly with the $q_k$ neighbourhood. The lower and upper face of $q_k$ has the same level of decomposition, being $n = 0$ and producing a single $q_{k+1}$ neighbour. On its right side, as the dimensions of the neighbour $q_{k+1}$ are larger than $q_k$, the number of neighbours is equal to 1, counting a total of 11 neighbours and possible movements in this case.

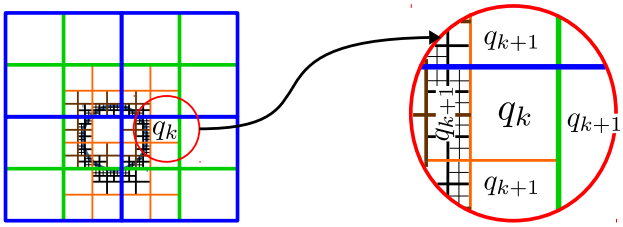



**Figura 7.3:** Neighbourhood structure. Quadtree decomposition neighbourhood.

The $3D$ decomposition methodology has two relevant variations. Firstly, the definition and location of each voxel boundary, and secondly, the definition of the number of neighbours per each voxel belonging to $S_{free}$. The number of neighbours $q_k$ is bounded by at least $q_{k+1} = 3$ neighbours, and the maximum number of each $q_k$ voxel face is a multiple of $(2^{0,1,2,\ldots,n})^2$ with $n$ decomposition levels (shown in Figure 7.4).

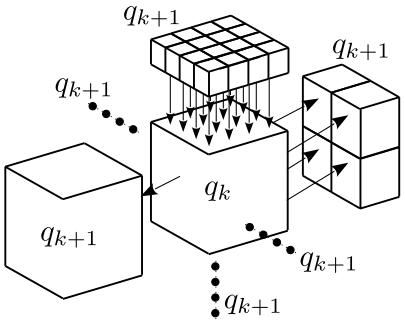



**Figura 7.4:** Neighbourhood structure. Octree decomposition neighbourhood.

At this point, the procedure is partially complete, since the decomposition just finds the set $S_{free}$ and additional calculations to determine if $\rho$ are needed. Hence, *MACD* uses *Dijkstra's* Dijkstra 1959 algorithm to calculate the optimal path in distance terms.

**Algorithm 5** Modified Adaptive Cell Decomposition (*MACD*)

```
1: n → decomposition level, nextRow = 0
2: [boundary] = environment.boundaries
3: rectangloid.add = boundary
4: for i = 0 : n do
5:   rows = rectangloid.size;
6:   nextRow = nextRow + 1;
7:   for j = nextRow : rows do
8:     if rectangloid(j).ocup == S_occup then
9:       [boundary] = boundaryOctree(rectangloid(j).boundaries);
10:      for k = 1 : boundary.size do
11:        newRow = newRow + 1;
12:        rectangloid(newRow).add = boundary(k, :);
13:        obj = obstacle.find ∈ rectangloid(newRow);
14:        if obj == FREE then
15:          rectangloid(newRow).ocup = S_free
16:        else
17:          rectangloid(newRow).ocup = S_occup
18:        end if
19:      end for
20:    end if
21:  end for
22:  nextRow = rows;
23: end for
24: vertex = rectangloid(:).boundary.center;
25: edges = rectangloid(:).neighbour.find;
26: ρ = Dijkstra(vertex, edges);
```

Algorithm 5 shows the pseudo-code to generate a structure called a *rectangloid* which contains all the information compiled throughout the cell decomposition process. The algorithm performs a recursive searching of the free $S_{free}$ space and the occupied $S_{occup}$ space in the discrete $3D$ environment to determine each voxel property (including its set of neighbours), and it simultaneously builds

the computational structure that joins the voxels. For a better understanding, a brief description of several algorithm steps is offered here.

**Line 9:** Boundaries of the current voxel are calculated.

**Line 12:** Boundaries of the sub-voxel are assigned to *rectangloid*.

**Line 13:** This step does a total routing by searching the environment for obstacle collisions.

**Line 24:** The *vertex* variable collects each ($S_{free}$) of *rectangloid* structure.

**Line 25:** The *edges* variable determines the structure that joins every *vertex*.

**Line 26:** *Dijkstra's* algorithm is used to determine $\rho$.

## 7.4 Recursive Rewarding Modified Approximate Cell Decomposition (*RR-MACD*)

In this section, a new algorithm for path planning in $3D$ environments is formulated so that using an external planner based on nodes, such as *Dijkstra* (Dijkstra 1959) or $A^*$ (Hart y Nils 1968), becomes unnecessary. Since it attempts to achieve a final path $\rho$ based on starting conditions, or initial states, each future system state is determined by the present one (each state is a collision-free neighbour voxel).

### *7.4.1 Methodology*

Let $\Upsilon$ denote a work environment as a discrete $3D$ space that contains a finite set of collision-free voxels ($S_{free}$) and a finite set of busy voxels ($S_{occup}$).

Let us assume an UAV is included within a collision-free voxel, which is considered as initial state $s_k$, with the aim of reaching the end point $q_f$. Let's assume a set formed by voxels of different sizes $S_{k+1}$ as a neighbourhood of $s_k$. In this context, a state model and a transition matrix can be developed as in Figure 7.5 to determine the optimal transition from $s_k$ to any state belonging to $S_{k+1}$ based on two transitional measurements ($D_1$ and $D_2$). Starting from the current state $s_k$, the method will try to obtain the optimum neighbour state belonging to $S_{k+1}$ ($s_k \rightarrow S_{k+1} = D_1$). It will then locate which sub-path from each neighbour in $S_{k+1}$ to the final point $q_f$ is best ($S_{k+1} \rightarrow q_f = D_2$).

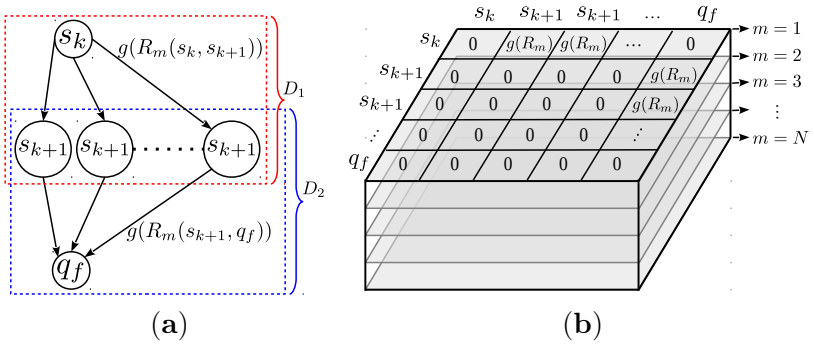



**Figura 7.5:** Generic structure state transition. (**a**) Generic state model. (**b**) Generic transition matrix.

To solve this problem a discrete deterministic finite automaton (DDFA) (Skoldstam, Akesson y Fabian 2007; Y.-H. E. Yang y Prasanna 2011), $F$, can be defined as $F = (S, G, D, q)^T$ with a set of $R_m$ partial functions, where:

- $q$ are two points in the $3D$ environment space, where
  - $q_i$ is the initial point
  - $q_f$ is the final point.
- $S$ is a finite set of $M$ current states, where
  - $S_{free}$ is the finite set of collision-free voxels. Split in the current voxel $s_k = [s_k(x), s_k(y), s_k(z)]$, and the set of its neighbours $s_{k+1} = [s_{k+1}(x), s_{k+1}(y), s_{k+1}(z)]$.
  - $S_{occup}$ is the finite set of occupied voxels.
- $R_m, m = 1 \dots N$ is a set of $N$ partial functions involved in the $3D$ UAV navigation characteristics and determining feasible progress. In this paper, $N = 4$ functions are defined as flight parameters, being:

$$\begin{aligned} R_1(i,j) &= \frac{M_{distance}(s_i \to s_j)}{M_{trDirect}} \in \mathbb{R} : [0,1] \\ M_{distance}(s_i \to s_j) &= \sqrt{(s_i - s_j))^2} \end{aligned} \tag{7.1}$$

where $M_{distance}(s_i \rightarrow s_j)$ is the Euclidean distance between any two states and $M_{trDirect}$ is the distance in a straight line between $q_i$ and $q_f$.

$$\begin{aligned} &R_2(i,j) = M_{tan}(s_i \rightarrow s_j) \in \mathbb{R} : [-1,1] \\ &M_{tan}(s_i \rightarrow s_j) = tan^{-1}\theta \\ &\theta = \left( \frac{\sqrt{[(s_i(y) - s_j(y)]^2 + [s_i(x) - s_j(x)]^2}}{s_i(z) - s_j(z)} \right) \end{aligned} \tag{7.2}$$

where $M_{tan}(s_i \rightarrow s_j)$ is the direction change measurement of the tangent vector to a curve, which shows the inclination angle between any two states.

$$\begin{aligned} &R_3(i,j) = M_{phi}(s_i \rightarrow s_j) \in \mathbb{R} : [-1,1] \\ &M_{phi}(s_i \rightarrow s_j) = \phi \\ &\phi = tan^{-1} \left( \frac{s_i(y) - s_j(y)}{s_i(x) - s_j(x)} \right) \end{aligned} \tag{7.3}$$

where $M_{phi}(s_i \rightarrow s_j)$ is the direction change of the bi-normal vector around the tangent vector between any two states.

$R_4(s_i, s_j)$ is associated with the amount of battery and determines the possibility of success on a predefined trajectory:

$$\begin{aligned} &R_4(i,j) = M_{batt}(s_i \rightarrow s_j) \in \mathbb{R} : [0,1] \\ &M_{batt}(s_i \rightarrow s_j) = \begin{cases} 0, \frac{batt_i \rightarrow batt_j}{Cur_{batt}} > Cur_{batt} \\ 1 - \frac{batt_i \rightarrow batt_j}{Cur_{batt}} \leq Cur_{batt} \end{cases} \end{aligned} \tag{7.4}$$

where $M_{batt}(s_i \rightarrow s_j)$ is the normalised theoretical quantity of battery needed to fly from any state to any other—and the $Cur_{batt}$ is the current amount of battery available for flight.

Further, a Gaussian function $g(R_m)$ is used to determine the reward in executing a possible action and it is defined as:

$$g(R_m) = \frac{\sin(\pi * R_m + \pi/2) + 1}{2} \tag{7.5}$$

where the transition cost values ($s_k \rightarrow S_{k+1}$, $S_{k+1} \rightarrow q_f$) have been normalised within boundaries $[0, 1]$. Notice how the greater the effort

$R_m$, the lower the reward $g(R_m)$ and vice-versa. Therefore, the execution of an action from state $s_i$ to different states $s_j$ produces state transitions at different costs—an elevated cost will produce a lower reward on the transition.

All these rewards can be expressed as a vector $G(i,j)$ of flight parameters such as:

$$G(i,j) = [g(R_1(i,j)), g(R_2(i,j)), \ldots, g(R_N(i,j))]^T \tag{7.6}$$

- $D \in \mathbb{R}^{M-2}$ is the received reward associated with a priority $\mathfrak{p} \in \mathbb{R}^N$ (i.e., $\mathfrak{p}$ assignes priorities to the defined functions $g(R_m)$, for example, if a better distance is required, then, $\mathfrak{p}$ associated to $R_1$ is higher than the rest of $R_m$, or, if lower battery consumption is required, then the $\mathfrak{p}$ associated to $R_4$ is higher) for executing an action on a function $g(R_m)$ and is stated as the sum of two transition priority vectors ($D_1$ and $D_2$) defined as:

$$\begin{aligned} D_1 =& (\mathfrak{p} \times G(i,j)) + \xi \\ & i = 1, j = 2 \ldots (M-1)] \end{aligned} \tag{7.7}$$

$$\begin{aligned} D_2 =& (\mathfrak{p} \times G(i,j)) + \xi \\ & [i = 2 \ldots (M-1), j = M] \end{aligned} \tag{7.8}$$

$$D = D_1 + D_2 \tag{7.9}$$

where $\xi$ is a predefined negative reward value in each state belonging to $S_{k+1}$. Notice that the probability distribution values of the functions $g(R_1), g(R_2), \ldots, g(R_N)$ are independent and the set of answer vectors $D$ which are mappings of $S \times S_{free}$. Therefore, this map is generated with a time-independent probability distribution. Hence, the probability of moving between one instant and the next does not change.

Hence, the best reward value from vector $D$ generates the best $x$—and the final path, denoted by $\rho_x(F)$, defines a finite labeled graph with vertex $S_x \in S_{free}$.

### *7.4.2 Simple Application*

To explain the methodology, and for sake of simplicity, let's state the problem of travelling from the actual state $q_i \in s_k$ or *init* point to the *goal* point $q_f$ in a $2D$ environment using a standard $2D$ cell decomposition. Figure 7.6a shows the initial scenario as well as the different $S_{k+1}$ states (observable and neighbours) that may become a new $s_k$. In this case, in $t = 0$, it is the same cost to move right or down—this situation appears due to the inherent symmetry of the decomposition methodology. In such a situation, randomness decides the next state.

Since a state cannot point to itself and can be visited only once, when the $S_{k+1}$ states are visited and evaluated, one of them is selected by producing a forwarding movement (see Figure 7.6(b)). Hence, the state selected is the new *initial* point $s_k$, and the process is repeated again (see Figure 7.6(c)). The network structure shown indicates a forward movement $s_k$ to the immediately next state $s_{k+1}$. This movement is independent of any previous state $s_k$.

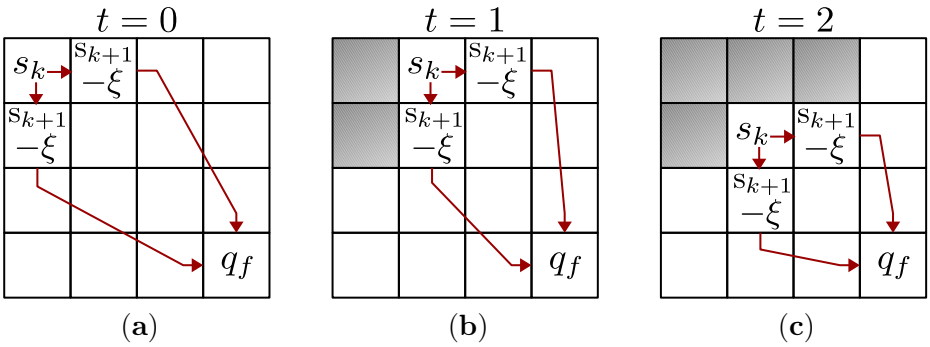



**Figura 7.6:** Network structure and state transition. (**a**) state transition $t = 0$. (**b**) state transition $t = 1$. **(c)** state transition $t = 2$.

Moving towards a $3D$ environment, a similar scene is represented as a master voxel containing an obstacle inside (Figure 7.7). This voxel can be split into different levels of voxels with different dimensional characteristics in different spatial locations. To obtain the finite set of collision-free spaces $S_{free}$, *MACD* is performed recursively.

The main environment boundaries have been defined from the initial coordinates $q_i \equiv (x_i, y_i, z_i)$ to the final one $q_f \equiv (x_f, y_f, z_f)$, resulting in a rectangular

shape, being $env = ([x_i, x_f], [y_i, y_f], [z_i, z_f])$. Each obstacle $h_i(x, y, z) \in \mathbb{R}^3 \rightarrow (x, y, z) = \lambda$, is defined as:

$$h_i(\lambda)|_t = h_i(\lambda)|_{t+1} \Rightarrow static \tag{7.10}$$

$$h_i(\lambda)|_t \neq h_i(\lambda)|_{t+1} \Rightarrow dynamic \tag{7.11}$$

where $h_i(\lambda) \rightarrow i > 0$ could take two possible states, *static* (Equation (7.10), the obstacle does not change its position with passing time) or *dynamic* (Equation (7.11), the obstacle changes its position to another with passing time).

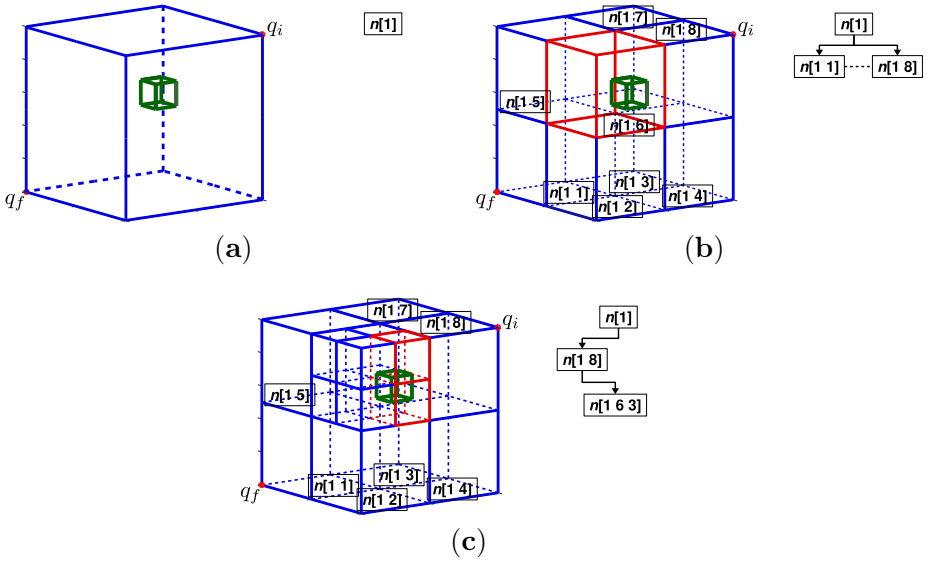



**Figura 7.7:** Recursive rewarding *MACD* (*RR-MACD*) start process example. (**a**) Complete environment. (**b**) First *MACD* level $n[1]$. (**c**) second *MACD* Level $n[1\ 8]$.

Figure 7.7(a) shows the environment definition (blue lines) as the main node $n[1]$ with an obstacle $h_1(\lambda)$ placed inside (box green lines). Once the first decomposition is performed (Figure 7.7(b)), a first octal level $n[[1\ 1], \ldots, [1\ 8]]$ is generated with nodes having different occupancy properties. Each will belong to $S_{free}$ if there is no $h_i(\lambda)$ within its voxel limits ($s_k \bigcap h_i(\lambda) = 0$) (blue lines), or to $S_{occup}$ if the voxel is partial or totally occupied by the obstacle ($(s_k \bigcap h_i(\lambda) = 1) \vee (s_k \in h_i(\lambda))$) (red lines).

The first step is to determine the $q_i$ container voxel (for this example, it is $n[1\ 8]$). In case of obstacle detection in $n[1\ 8]$, a recursive decomposition would be performed on the location. At this level, the node $n[1\ 8]$ is the new starting state $s_k$ and the observable neighbours $S_{k+1}$ are the nodes $n[1\ 4], n[1\ 6]$, and $n[1\ 7]$. At this stage, the state model is depicted in Figure 7.8 and the transition matrix is detailed in Table 7.3.

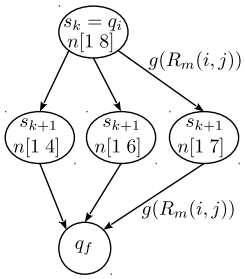



**Figura 7.8:** State model $M = 5$ states. List of states: $s_1 \rightarrow n[1\ 8]$, $s_2 \rightarrow n[1\ 4]$, $s_3 \rightarrow n[1\ 6]$, $s_4 \rightarrow n[1\ 7]$, $s_5 \rightarrow [q_f]$.

The probability distribution for the discrete states can be derived from a multidimensional matrix of dimensions $M \times M \times N$ (see Table 7.3) where, $M$ is the number of states, and $N$ is the number of partial functions $R_{(m=1\ldots N)}$ involved in the $3D$ UAV navigation.

**Tabla 7.3:** Initial multidimensional transition matrix listed by first column ($i = 1 \ldots M$) and the first row ($j = 1 \ldots M$). The number of levels are given by the number of $N$ functions $R_m$ for the particular navigation characteristics.

$\boldsymbol{m=\ N} \rightarrow$
$\boldsymbol{m=\ \cdots} \rightarrow$
$\boldsymbol{m=3} \rightarrow$
$\boldsymbol{m=2} \rightarrow$

| | $\boldsymbol{m=1} \rightarrow$ | $\boldsymbol{j=1}$ | $\boldsymbol{j=2}$ | $\boldsymbol{j=3}$ | $\boldsymbol{j=4}$ | $\boldsymbol{j=M}$ |
|---|---|---|---|---|---|---|
| | | $s_k = q_i$ | $s_{k+1}$ $n[1\ 4]$ | $s_{k+1}$ $n[1\ 6]$ | $s_{k+1}$ $n[1\ 7]$ | $q_f$ |
| $\boldsymbol{i=1}$ | $s_k = q_i$ | 0 | $g(R_m)$ | $g(R_m)$ | $g(R_m)$ | 0 |
| $\boldsymbol{i=2}$ | $s_{k+1}$ $n[1\ 4]$ | 0 | 0 | 0 | 0 | $g(R_m)$ |
| $\boldsymbol{i=3}$ | $s_{k+1}$ $n[1\ 6]$ | 0 | 0 | 0 | 0 | $g(R_m)$ |
| $\boldsymbol{i=4}$ | $s_{k+1}$ $n[1\ 7]$ | 0 | 0 | 0 | 0 | $g(R_m)$ |
| $\boldsymbol{i=M}$ | $q_f$ | 0 | 0 | 0 | 0 | 0 |

This multidimensional matrix equivalent to Figure 7.5(b) has been constructed with the information of the state model and the reward values. The aim is to find the partial responses coming from the first row and last column, and split in two transition priority vectors $(D_1, D_2) \in \mathbb{R}^{(M-2)}$. Using the priority vector $\mathfrak{p}$ and the offset $\xi$, vectors $D_1$ and $D_2$ deliver partial reward distributions to a possible next state.

The transition priority vector $D_1$ is built with the set of columns $j = 2 \dots (M-1)$ and the row $i = 1$ such that

$$
\begin{aligned}
D_1 &= \left[\mathfrak{p} \times G(i,j)\right] + \xi, i = 1, j = 2 \dots (M-1) \\
G(1,2) &= \begin{bmatrix} g(R_1(1,2)) \\ g(R_2(1,2)) \\ \vdots \\ g(R_N(1,2)) \end{bmatrix} \\
G(1,3) &= \begin{bmatrix} g(R_1(1,3)) \\ g(R_2(1,3)) \\ \vdots \\ g(R_N(1,3)) \end{bmatrix} \\
&\vdots \\
G(1,(M-1)) &= \begin{bmatrix} g(R_1(1,(M-1))) \\ g(R_2(1,(M-1))) \\ \vdots \\ g(R_N(1,(M-1))) \end{bmatrix}
\end{aligned}
\tag{7.12}
$$

The second transition priority vector, $D_2$, is built with the set of rows $i = 2 \dots (M-1)$ and column $j = M$.

$$
\begin{aligned}
D_2 = \left[\mathfrak{p} \times G(i,j)\right] + \xi, i = 2 \ldots (M-1), j = M \\
G(2,M) = \begin{bmatrix} g(R_1(2,M)) \\ g(R_2(2,M)) \\ \vdots \\ g(R_N(2,M)) \end{bmatrix} \\
G(3,M) = \begin{bmatrix} g(R_1(3,M)) \\ g(R_2(3,M)) \\ \vdots \\ g(R_N(3,M)) \end{bmatrix} \\
\vdots \\
G((M-1),M) = \begin{bmatrix} g(R_1((M-1),M)) \\ g(R_2((M-1),M)) \\ \vdots \\ g(R_N((M-1),M)) \end{bmatrix}
\end{aligned}
\tag{7.13}
$$

Finally, the final reward vector $D$ expressed as the sum of $D_1$ and $D_2$ contains the optimal value which points to the best state (node) for continuing the search of the path $\rho_x$:

$$D = D_1 + D_2 \tag{7.14}$$

$$x = best(D) \tag{7.15}$$

To continue with the example in Figure 7.7, let us assume that $x = best(D)$ points to $n[1\ 6]$. Nevertheless, $n[1, 6]$ is occupied (it belongs $S_{occup}$), so $MACD$ is invoked, creating a new level in the data structure, composed of $n[1\ 6\ (1 \ldots 8)]$ (see Figure 7.7(c)). Even though the state $s_k$ remains in $n[1\ 8]$, the new decomposition on $n[1\ 6]$ returns a new set of neighbours, which join with previous ones, and define the new set $S_{k+1}$ defined by ($n[1\ 4]$, $n[1\ 6\ 3]$, $n[1\ 6\ 4]$, $n[1\ 6\ 7]$, $n[1\ 6\ 8]$, $n[1\ 7]$).

So looking for the optimum within $S_{k+1}$ is required. Let us assume the best node from $s_{k+1}$ is $n[1\ 6\ 3]$ (notice that $MACD$ is invoked once until now) and so the new state $s_k$ is reassigned to $n[1\ 6\ 3]$ and consequently, the neighbourhood of the new state $s_k$, is conformed by $S_{k+1} = \{n[1\ 6\ 1]$, $n[1\ 6\ 4]$, $n[1\ 6\ 7]$, $n[1\ 5]$, $n[1\ 8]\}$.

The previous actions produce the displacement from a current $s_k$ state to the next best state, and towards the final point. While the container voxel of the resulting better state $x$ does not contain the *goal* point, there is the possibility that $x$ has neighbours in different decomposition levels. Hence, the process continues until the *goal* point is reached.

Once the previous phase has been completed, the optimal path $\rho_x(F)$ is totally determined and the search finishes. The flowchart depicted in Figure 7.9 and the Algorithm 6 show the pseudo-code for the described actions.

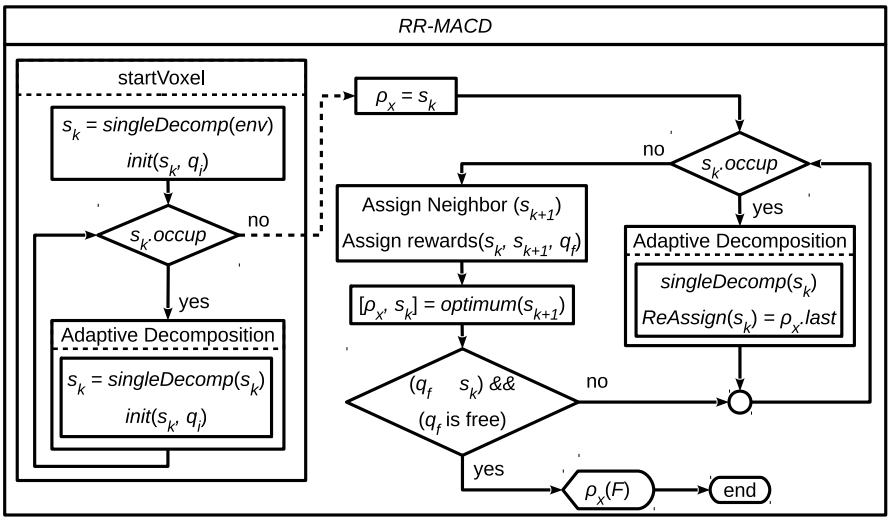



**Figura 7.9:** Flowchart of *RR-MACD*. Notice how steps 3 to 8 in Figure 7.2 are re-used in this chart and renamed as *singleDecomp*

The procedure is split in two stages. Firstly, a start voxel location is defined and, if there is an $h_i(\lambda)$ in the environment, an initial simple decomposition *singleDecomp* is performed (as a result, the collision-free voxel that contains the *init* point will be defined). Once the initial $s_k$ state is assigned—which is also an initial $\rho_x$—the procedure proceeds depending on its occupation. If $s_k$ is collision-free, the neighbours $S_{k+1}$ are assigned, the rewards are calculated, and the optimal $x$ is located. Therefore, the best state $x$ becomes the new $s_k$ and is added to $\rho_x$. If this new $s_k$ is occupied, the process requires another decomposition on the current $s_k$, and $s_k$ will return to its previous state. The procedure is completed when the current $s_k$ contains the *goal* point and $s_k$ is collision-free.

Algorithm 6 summarises the procedure described in Figure 7.9. In line 2, a search of the first containing $q_i \in s_k$ is performed. The loop continues until the current $s_k$ state contains $q_f$ (line 3). If the current state $s_k$ collides with an obstacle $h_i(\lambda)$, the $s_k$ state is decomposed (line 5: *singleDecomp*) and $s_k$ returns to its previous state (line 6). In line 8, the neighbours of $s_k$, $S_{k+1}$, are defined. Line 9 measures the transition rewards for any neighbour in $S_{k+1}$. Finally, the optimum is added to the path $\rho_x$ and $x$ is assigned as the new $s_k$ in line 10. When the loop finishes, the complete path $\rho_x(F)$ is returned (line 13).

**Algorithm 6** *RR-MACD*

```
1: define targetPoints, Obstacles
2: [s_k] = startVoxel(env)
3: while q_f ∈ s_k do
4:   if s_k.occup == true then
5:     singleDecomp(s_k);
6:     s_k = ρ_x.last;
7:   else
8:     [S_{k+1}] = neighbourhood(s_k);
9:     [D_1, D_2] = rewards(s_k, S_{k+1}, q_f);
10:    [ρ_x, s_k] = D.optimum
11:  end if
12: end while
13: return ρ_x(F)
```

The methodology described presents the following properties:

(a) A stochastic process in discrete time has been defined (it lacks memory), the probability distribution for a future state depends solely on its present values and is independent of the current state history.

(b) The sum of the priorities defined in vector $\mathfrak{p}$ is not equal to 1. $\sum_{i=1}^{N} \mathfrak{p}_i \neq 1$.

(c) The sum of the values of each priority vector, is not equal to 1. $\sum D_1 \neq 1$ and $\sum D_2 \neq 1$.

Finally, it should be noted that the environmental discrete decomposition results in ever smaller voxels of differing sizes. The level of voxel decomposition is variable based on two goals that must be fulfilled: (1) Designer defines the maximum decomposition level (minimum voxel size); however, the algorithm

tries to reduce the computational cost and, as a consequence, the minimum voxel size is generally avoided. (2) As soon as a free space meets the defined constraints it is selected by the algorithm regardless of the size of the voxel.

### *7.4.3 Dynamic environment approach*

The *RR-MACD* can be applied to a $3D$ UAV environment with obstacles for movement. Once $q_i$ and $q_f$ points are defined, the trajectory $\rho_x(F)$ is calculated and the UAV navigates through it. A dynamic environment implies a positional Equation (7.11). If the obstacles intersect the previously calculated trajectory $(h_i(\lambda)|_t \bigcap \rho_x(F) = 1)$, a new trajectory must be generated. The described methodology in the previous sections has constant targets $q_i$ and $q_f$. However, for a dynamic trajectory the path must be updated with a new $q_i$ in the current UAV location.

## 7.5 Experiments

In this section, a comparison of computational performance between *MACD* and *RR-MACD* algorithms is made. Three different simulation examples have been carried out with 10 executions of each algorithm over each environment. The 150 set of responses supplies the results to determine the performance. The algorithms have been run in a "8 x Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz" computer (Manufacturer: Gigabyte Technology Co., Ltd., Model: B85M-D3H) with 8Gb RAM and S.O. Ubuntu Linux 16.04 LTS. The algorithms were programmed in MATLAB version 9.4.0.813654 (R2018a).

For each simulation example, five different 3D environments were defined. Table 7.4 shows one row per environment, where the column entitled *"UAV target coordinates"* expresses the coordinates in terms of *init* and *goal* points. The environmental dimensions have been defined in distances related to those coordinates (in a scale of *meters* $m$). The column *"Obstacles Dimensions"* shows the dimensional characteristics of each obstacle and the *"Obstacles Location"* places the obstacles in a specific location. Moreover, the altitude between *init* and *goal* target points are different, guaranteeing that the UAV path will be built in the $(x, y, z)$ axes. It should be highlighted that maps are created with predefined static obstacles for the different groups of experiments.

**Tabla 7.4:** Definition of five different 3D environments for the simulation examples.

| # | ***UAV Target Coordinates (m)*** | | | | | | ***Obstacles Dimensions (m)*** | | | ***Obstacles Location (m)*** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***init*** | | | ***goal*** | | | | | | | | |
| | ***x*** | ***y*** | ***z*** | ***x*** | ***y*** | ***z*** | ***x*** | ***y*** | ***z*** | ***x*** | ***y*** | ***z*** |
| **1** | 100 | 100 | 42 | 0 | 0 | 24 | 12 | 12 | 50 | 40 | 30 | 25 |
| | | | | | | | 15 | 15 | 30 | 24 | 40 | 15 |
| | | | | | | | 30 | 30 | 30 | 70 | 20 | 15 |
| | | | | | | | 15 | 15 | 46 | 20 | 70 | 23 |
| | | | | | | | 12 | 12 | 54 | 80 | 70 | 27 |
| **2** | 1000 | 1000 | 600 | 0 | 0 | 420 | 200 | 200 | 200 | 333 | 333 | 333 |
| | | | | | | | 300 | 300 | 300 | 777 | 777 | 777 |
| **3** | 1000 | 1000 | 300 | 0 | 0 | 700 | 100 | 100 | 100 | 400 | 400 | 400 |
| | | | | | | | 150 | 150 | 150 | 400 | 400 | 800 |
| **4** | 1200 | 1200 | 390 | 0 | 0 | 720 | 200 | 300 | 400 | 200 | 800 | 400 |
| | | | | | | | 20 | 20 | 20 | 300 | 200 | 700 |
| **5** | 1200 | 1200 | 800 | 0 | 0 | 500 | 10 | 10 | 10 | 600 | 600 | 600 |
| | | | | | | | 15 | 15 | 15 | 200 | 800 | 800 |
| | | | | | | | 15 | 15 | 15 | 200 | 800 | 200 |

First, an urban environment with several buildings is described. For the construction of this environment, a maximum altitude reference has been taken, such as stated in "The Regulation of Drones in Spain 2019 (*Normativa Sobre Drones en España [2019] - Aerial Insights* s.f.; *Disposición 15721 del BOE núm. 316 de 2017 - BOE.es* s.f.). For environments defined as $2, 3, 4$, and $5$, larger dimensions have been considered in which a varying number of obstacles in different air spaces are defined.

### *7.5.1 Example 1. Static Obstacles and Four Flight Parameters (constraints)*

This example shows the performance of *RR-MACD* when four functions $R_m$ are used ($N = 4$). These functions are the same as those detailed in Section 7.4. In Figure 7.10, the specifically results for environments #1 and #2 are shown. Therefore, Figure 7.10(a) shows the urban environment. Hence, Figure 7.10(b) shows the built path by *RR-MACD* of the environment #1, where black boxes are the obstacles $h_i(\lambda)$, green stars shows the voxel set of vertices in the state $s_k$ and the orange line shows the final path $\rho_x(F)$.

In Figure 7.10(c), the resulting *MACD* technique on the environment #2 can be appreciated, where black boxes are the static obstacles $h_i(\lambda)$, blue boxes are the set of voxels used for the *Dijkstra's* planner to obtain the optimal final path (green stars are the vertices set of its belonging voxel and the orange line shows the final path $\rho$. Finally, Figure 7.10(d) depicts the final trajectory built by *RR-MACD*, where the orange line shows the final path $\rho_x(F)$.

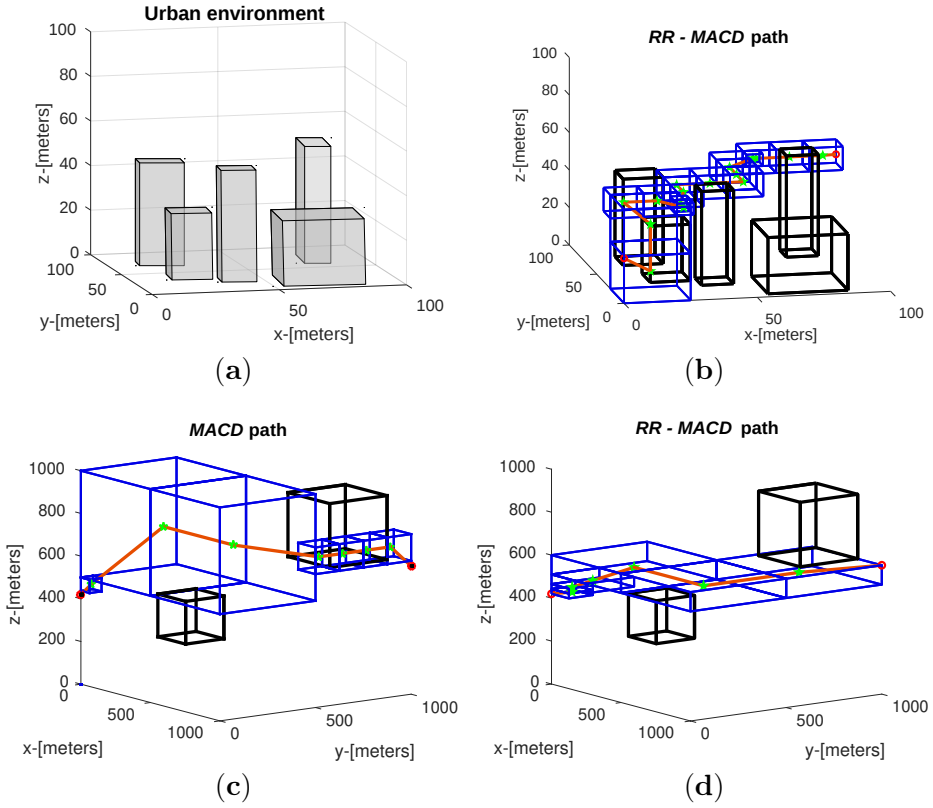



**Figura 7.10:** Graphical Results of Example 1. (**a**) Modeled urban environment cluttered with surrounding buildings. (**b**) Results for environment #1 with *RR-MACD*. (**c**) Final path generated using *MACD* for environment #2. (**d**) Final path generated using *RR-MACD* for environment #2

Comparisons of both algorithms regarding computational time is shown in Table 7.5. The results show an important advantage in decomposition time $S_{free}$ and $\rho_x(F)$ spaces for *RR-MACD*. Notice that, when an obstacle $h_i(\lambda)$

intersects with a voxel decomposition, *MACD* recursively continues until the predefined level $n$ (for this reason in environment #4 the searching time for *MACD* is considerably greater than other environments).

**Tabla 7.5:** Comparison of both algorithm executions for different environments. Column recursive rewarding modified adaptive cell decomposition (*RR-MACD*) 4 vs. *MACD* (%) shows the average resources (decomposition time (s), number of free voxel decomposition "$S_{free}$" and number of nodes in the final path "$\rho$") used for *RR-MACD* in comparison with *MACD*. Column *RR-MACD* 10 vs. *RR-MACD* 4 (%) shows the average resources (decomposition time (s), number of free voxels decomposition "$S_{free}$" and number of nodes in the final path) used for *RR-MACD* when the number of flight parameters (constraints) are augmented to 10.

| # | ***MACD*** | | | | ***RR-MACD 4 Constraints*** | | | ***RR-MACD 4 vs. MACD (%)*** | | | ***RR-MACD 10 vs. RR-MACD 4 (%)*** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **dec. Time (s)** | **Dijks. Time (s)** | **# $S_{free}$** | **# $\rho$** | **dec. Time (s).** | **# $S_{free}$** | **# $\rho_x(F)$** | **dec. Time (s)** | **$S_{free}$** | **$\rho$** | **dec. Time** | **$S_{free}$** | **$\rho$** |
| **1** | 0.117 | 0.038 | 205 | 19 | 0.056 | 115 | 18 | 36.238 | 54.641 | 93.684 | +51.449 | +74.975 | +50.000 |
| **2** | 0.104 | 0.049 | 496 | 11 | 0.012 | 27 | 8 | 8.368 | 5.443 | 72.727 | +18.710 | +26.337 | +25.000 |
| **3** | 0.151 | 0.048 | 426 | 13 | 0.014 | 19 | 6 | 7.105 | 4.460 | 46.153 | −25.118 | −18.723 | +1.851 |
| **4** | 3.535 | 1.021 | 5201 | 19 | 0.003 | 11 | 6 | 0.080 | 0.211 | 31.578 | +327.894 | +363.636 | +57.407 |
| **5** | 0.078 | 0.032 | 294 | 23 | 0.009 | 19 | 7 | 8.470 | 6.462 | 30.434 | +115.017 | +79.532 | +33.333 |

The third column *"RR-MACD 4 vs. MACD (%)"* shows in percentages the differences between *MACD* and *RR-MACD 4* regarding environmental decomposition time, number of $S_{free}$ free-spaces generated during the searching process, and the number of final path nodes $\rho$. It is important to mention that *MACD* needs an additional time because *Dijks time(s)* shows the seconds needed to find $\rho$. For example, in the first environment, the *RR-MACD* algorithm just needs 36.238 % of the time that *MACD* takes for environment decomposition, and 54.641 % of the voxels that *MACD* takes to generate $S_{free}$, and 93.684 % of the nodes that *MACD* needs to build $\rho$. Therefore, *RR-MACD* shows a general improvement on the process.

### *7.5.2 Example 2. Static Obstacles and 10 Constraints*

In example 1, the set of partial functions involved in $3D$ UAV navigation is equal to 4. For this example, an additional set of 6 random values were added as new functions to simulate complex flight characteristics. Therefore, the number of partial functions in $3D$ UAV navigation $R_m$ will be equal to $M = 10$ and the probability distribution multidimensional matrix now has 10 levels. This increment of functions, and its random nature, will provoke inherent changes in the results. These can be observed in the fourth column of Table 7.5, entitled *"RR-MACD 10 vs. RR-MACD 4 %"*, where the relative difference between *RR-*

*MACD 4* and *RR-MACD 10* shows the percentage increase or decrease in decomposition time, $S_{free}$ and $\rho_x(F)$.

For example, let us compare performances between *RR-MACD 10* and *RR-MACD 4* in the environment #1. *RR-MACD 10* needs 51.499 % more time to find a final path. It generates 74.975 % plus voxels and the number of nodes in the final path $\rho$ is 50 % higher.

Table 7.6 shows a set of additional data corresponding to the results in terms of distances travelled between the *init* point and the *goal* point after the execution of each algorithm. As mentioned in the previous sections, the main objective of planning is not to reach optimality in exclusive terms of distance.

**Tabla 7.6:** Path results in distance metrics.

| | **Distance Travelled Meters (m)** | | |
|---|---|---|---|
| **Scene** | ***MACD*** | ***RR-MACD*** 4 | ***RR-MACD*** 10 |
| 1 | 224.060 | 197.410 | 241.600 |
| 2 | 1853.000 | 1592.545 | 1734.054 |
| 3 | 1768.600 | 1790.181 | 1728.300 |
| 4 | 1693.100 | 1868.463 | 2221.354 |
| 5 | 1731.800 | 1829.690 | 2123.954 |

### *7.5.3 Example 3. Dynamic Obstacles*

An additional experiment was performed that considered obstacles in motion. A new environment has been proposed for obstacles intersecting with the calculated $\rho_x(F)$. Hence, a new $\rho_x(F)$ with a new *init* (actual location of the UAV) is built.

Figure 7.11 represents this new environment with two obstacles in motion (black boxes). The first dynamic obstacle begins its displacement in location (600, 600, 600)m, and performs a continuous motion in an *east* direction with a constant velocity of 15 m/s. The second one begins its flight in location (80, 800, 600)m with a constant velocity of 15 m/s in the same direction. After 11 s, the first obstacle collides with $\rho_x(F)$ (this crash is illustrated with orange line in Figure 7.11(a) and a new $\rho_x(F)$ is calculated, Figure 7.11(b) shows the new location of the obstacles (notice that limits in $x$ and $y$ axis have changed from 1000 m to 800 m). At $t = 34$ s, a new collision is detected (Figure 7.11(c)), even though the first obstacle is out of the environment, the second one has

collided with $\rho_x$. When the UAV (blue square) has passed this part of $\rho_x(F)$, the new trajectory until *goal* is collision-free.

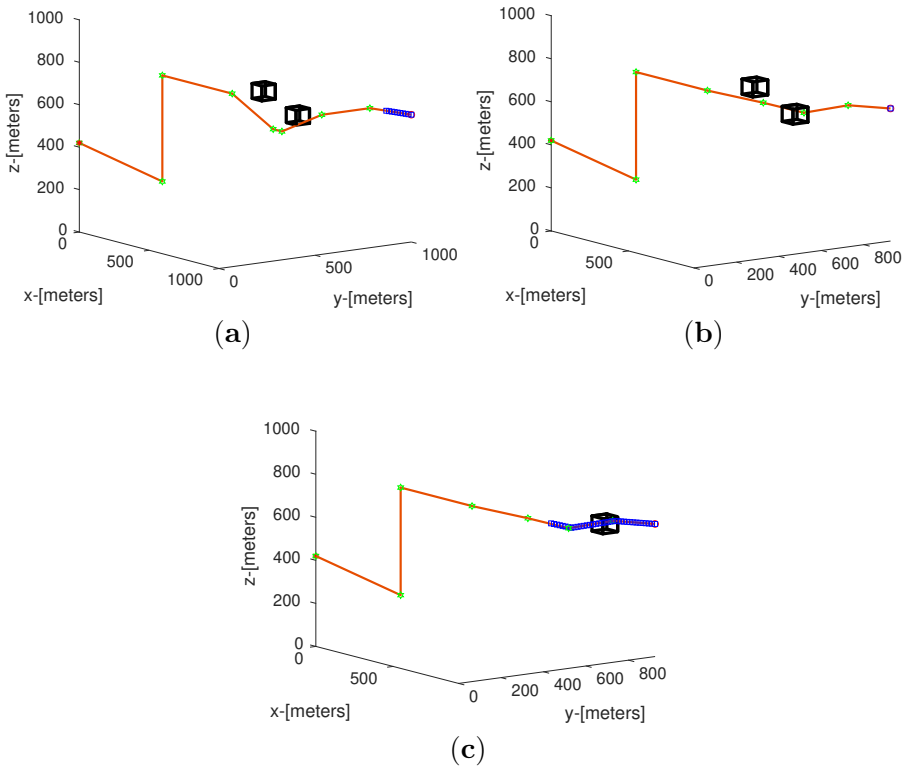



**Figura 7.11:** Network structure and state transition. (**a**) Collision in time of flight $t = 11s$. (**b**) New $\rho_x(F)$ after first collision $t = 11s$. (**c**) The new trajectory $\rho_x(F)$ does not change after the second collision ($t = 34s$) since the collision comes on the trajectory surpassed by the UAV.

## 7.6 Conclusions

This paper presents an adaptive grid methodology in $3D$ environments applied to flight path planning. The approach described considers different constraints such as UAV maneuverability and geometry or static and dynamic environment obstacles.

The proposed algorithm, *RR-MACD*, divides the $3D$ environment in a synthesised way and does not need to invoke any additional planner to search (such as *Dijkstra* or $A^*$) for an optimal path generation. The improvement in the computational effort and the reduction in the number of nodes generated by the *RR-MACD* has been shown. The stochastic process in discrete time involved in the algorithm also shows a future probability distribution that only depends on its present states.

The partition of the $3D$ space into a defined geometric form enables decreasing the number of control points in the generated trajectory. In addition, the issue of computational cost and complexity has been addressed by providing a solution that generates relatively shorter time responses compared to techniques for generating similar trajectories.

In a future work, an additional processing task will be carried out, using the set of nodes, or *control points* $\rho_x(F)$ generated, to create a smoother path—and then the methodology will be tested under real flight conditions on an UAV model as in (Velasco-Carrau y col. 2016; Vanegas y col. 2018; Samaniego, Sanchis, Garcia-Nieto y col. 2018)

## Capítulo 8

# Comparative Study of 3-Dimensional Path Planning Methods Constrained by the Maneuverability of Unmanned Aerial Vehicles

**Nota sobre el capítulo:** En este capítulo se divide el espacio de trabajo 3D en dos planos 2D, y sobre cada plano se crea una curva continua 2D. Estas curvas son interpoladas y unidas con el objetivo de construir una tercera curva continua 3D. El contenido de este capítulo aparece en el siguiente trabajo:

> Samaniego, F., Sanchis, J., García-Nieto, S., & Simarro, R. (2018, October). *Comparative Study of 3-Dimensional Path Planning Methods Constrained by the Maneuverability of Unmanned Aerial Vehicles*. In 2018 7th International Conference on Systems and Control (ICSC) (pp. 13-20). IEEE.

***Abstract:*** *This paper presents a comparison of two classic path planning techniques, Rapidly-Exploring Random Tree and Modified Adaptive Cell Decomposition applied to 3-Dimensional space path planning generation for UAVs. The study presented focuses on the inclusion of real fixed-wing UAVs maneuverability, where the dynamic and kinematic behaviour introduces hard constraints in real environments. In addition, the inclusion of Dubins and Clotoids curves in 3-Dimensional spaces has been considered in terms of improving the results of these classical methods. Flight simulation results have been included to clarify this comparative study.*

## 8.1 Introduction

A well-defined challenge in flight planning algorithms is to incorporate active obstacle avoidance strategies to ensure a continuous, smooth and safe airworthiness. This challenge involves taking into account several aspects such as tangential accelerations, partial derivatives of their dimensional components, odometry, noise, etc. For that reason, it is a hard-working and active field for research.

Since plenty and different methodologies have been proposed to fulfill the task of path planning and obstacle avoidance, two of the most relevant approaches in the literature have been selected for comparison in this work. On the one hand, Rapidly-Exploring Random Tree (RRT) for continuous space mapping (Melchior y Simmons 2007; Steven M LaValle 1998; LaValle, Steven M and Kuffner Jr 2000) and, on the other hand, Modified Adaptive Cell Decomposition (MACD) based on discretization space techniques (Borgefors 1986; Samet y Kochut 2002).

The general approach for the main techniques referenced above is to describe the optimal path as a set of discrete points. This set of multi-dimensional points, in general, are not traceable for a Unmanned Aerial Vehicle (UAV) with non-holonomic (Fixed wing UAVs are a non-holonomic particular case) characteristics (Justh, Eric W and Krishnaprasad 2002). UAVs with this particular structure perform a continuous flight at a defined speed, their movements are constrained by their maneuverability. Therefore, any calculated flight planning must meet the constraints introduced by the dynamic model of the UAV.

The classic hypothesis for UAVs obstacles avoidance analysis is performed in the 2-Dimensional or 2.5-Dimensional space (Kwangjin Yang y Sukkarieh 2008; Peng Cheng, Keller y V. Kumar 2008). This fact implies an indirect rejection of possible maneuvers on the altitude axis. In other terms, the main strategy to avoid obstacles is lateral maneuvers. However, the 3-Dimensional characteristics of the obstacle have to be taken into account. Therefore, an interesting approach might be to search for trajectories that can go around the obstacle not only on its sides but also above or below it. For this reason, the work presented, extends the general hypothesis to 3-Dimensional space to exploit the full maneuverability capabilities of UAVs.

This paper focuses on three objectives, first of all, two different algorithms, described in the literature, for $3D$ (continuous or discrete space) path planning will be compared. Second, an additional method to create $3D$ smooth paths, taking into account the dynamic model of UAVs, will be introduced (Isaac J Schoenberg 1988). Finally, the best planner analyzed will be used to build the guidance law, for known or unknown environments, for UAVs (see Fig. 8.1).

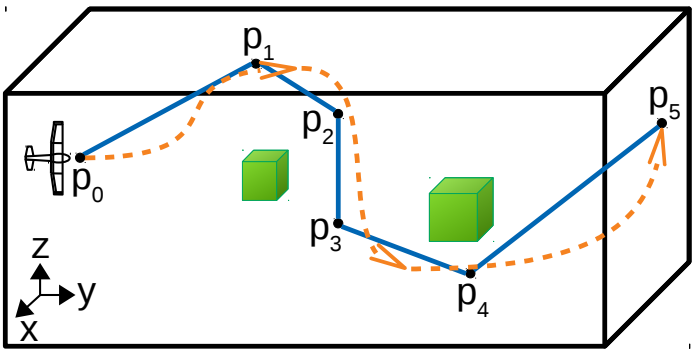



**Figura 8.1:** UAV description of motion problem.

The paper is organized as follows. In section 8.2: UAV model and curvature approach is defined. In section 8.3, a brief comparison of two different route planning algorithms is discussed. In section 8.4, a new approach to generating smooth paths is presented. In section 8.5, the results are analyzed through simulation and statistical data, using flight simulation models. Finally, main conclusions and future works are pointed out.

## 8.2 UAV Description model

### *8.2.1 UAV non-holonomic System*

Consider $\mathfrak{B}$ a non-holonomic UAV of mass $\mathfrak{M}$ and inertia $\mathfrak{I}$ that displaces in a space $\mathbb{R}^3$, the state space for this UAV model and its displacement environment is given by

$$\begin{aligned} q_R(t) =&(X_R(t), Y_R(t), Z_R(t), \\ &F_R(t), G_R(t), V_R(t), W_R(t))^T \in \mathbb{R}^7 \end{aligned} \tag{8.1}$$

where $X_R(t)$, $Y_R(t)$ and $Z_R(t)$ represent the Tait-Bryan conversion system (Weisstein 2009; Diebel 2006) with common origin. If the time dependence is removed and ideal conditions are assumed for the UAV to perform a stationary flight, then the total force $F_R(t) = 0$, total moment $G_R(t) = 0$, linear velocity and angular velocity are constant $V_R(t) = W_R(t) = k$, a reduced configuration can be written as

$$\dot{q}_R = [\phi_R, \theta_R, \psi_R]^T \tag{8.2}$$

where $\psi_R(t)$ is the yaw angle, $\theta(t)$ is the pitch angle and $\phi_R(t)$ is roll angle of the aircraft.

Assuming refractions in a Flat Earth Approximation geometry model (Young 2006), the local UAV system remains aligned to an objective Cartesian system $q_T = [X_T, Y_T, Z_T]^T$ can be assumed. Therefore, the system equations match the points of each model. Hence connects at all times the generated points by the UAV position $\dot{q}_R$ to another equivalent derived configuration $q_T$.

The relation with Cartesian coordinates on open sets are: $U = \{(r, \theta, \varphi) | r < 0, 0 < \theta < \pi, 0 \leq \varphi < 2\pi\}$ and $V = \{(x, y, z) | x^2 + y^2 + z^2 > 0\}$.

Exist an univoc corresponding $F' : V \to U$ between Cartesian and Spherical coordinates, where the singular extends to the $z$ axis, with $x^2 + y^2 = 0$, where $\varphi$, is not defined. Besides, $\varphi$ is not continuous in $(x, y, z)$ such that $x = 0$.

Hence, the inverse function $F^{-1}$ between the same open sets can be described in terms of:

$$\begin{aligned} x &= r \sin\theta \cos\varphi \\ y &= r \sin\theta \sin\varphi \\ z &= r \cos\theta \end{aligned} \tag{8.3}$$

The work presented has been tested using the UAV *kadett 2400*, its model has been described in (Velasco-Carrau y col. 2016). The aircraft has a very lightweight frame and characteristics that make it suitable for the purposes of this research. These characteristics include a 2,4 m wingspan, $0{,}9m^2$ of wing surface, $48{,}07Nm^2$ wing loading, and $1{,}65 \times 102m^3$ of available volume.

The UAV three aerodynamic components, roll $p$, yaw $r$ and pitch $q$ are modified by the control surfaces, *(ailerons, elevators* and, *rudder)*, which deflections are $\delta_a, \delta_r, \delta_e$ respectively and a body reference frame $X_b, Y_b, Z_b$ depicted in Fig. 8.2.

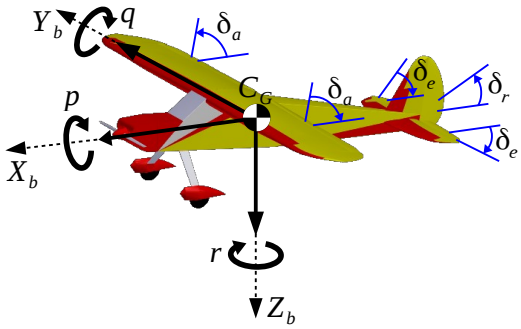



**Figura 8.2:** UAV *kadett 2400* simulation model.

Therefore, the flight simulation focuses on a local control stage in $\delta_a$, $\delta_r$ and $\delta_e$ which determines vertical and horizontal maneuvers. On the other hand, in the same local control stage, a throttle control which determines a continuous velocities variation has been done.

### *8.2.2 Curvature Radius*

The section 8.2, describes the UAV moving, but the curvature capability has not been defined. The curvature radius is the magnitude that measures the direction change of the tangent vector curve to a differentiable object embedded on Euclidean space (Toponogov 2006).

A parametric equation $\lambda(t) : \mathbb{R} \to \mathbb{R}^n$ from which it is attempted to deduce its curvature radius $\rho$. Whether the parametric circumference equation $g : \mathbb{R} \to \mathbb{R}^n$ assume the same value of $\lambda$ and satisfy $g'(t) = \lambda'(t)$ and $g''(t) = \lambda''(t)$ to each fixed $t$. Hence, the radius does not depend on the position of $\lambda(t)$, solely to the velocity $\lambda'(t)$ and the acceleration $\lambda''(t)$. Therefore, $\rho$ depends only on the scalars $|\lambda'(t)|^2, |\lambda''(t)|^2$ and $\lambda' \cdot \lambda''(t)$.

Since the circumference general parametric equation $g(u) = A\cos(h(u)) + B\sin(h(u)) + C$, where $C \in \mathbb{R}^n$ is the center, $A, B \in \mathbb{R}$ are perpendicular vectors of $\rho$ module. Hence, $A \cdot A = B \cdot B = \rho^2 \wedge A \cdot B = 0$, $h : \mathbb{R} \to \mathbb{R}$ is a doubly differentiable function in $t$. The derivative system is described as

$$\begin{aligned} |g'|^2 &= \rho^2 (h')^2 \\ g' \cdot g'' &= \rho^2 h' h'' \\ |g''|^2 &= \rho^2((h')^4) + (h'')^2 \end{aligned} \tag{8.4}$$

therefore, the derived system in relation to $\lambda$ is

$$\begin{aligned} |\lambda'^2(t)| &= \rho^2 h'^2(t) \\ \rho'(t) \cdot \lambda''(t) &= \rho^2 h'(t) h''(t) \\ |\lambda''^2(t)| &= \rho(h'^4(t) + h''^2(t)) \end{aligned} \tag{8.5}$$

the resulting system in $\rho, h'(t)$ and $h''(t)$ returns

$$\rho = \frac{|\lambda|^3}{\sqrt{|\lambda'|^2 |\lambda''|^2 - (\lambda' \cdot \lambda'')^2}} \tag{8.6}$$

where $\rho$ is the curvature radius geometric magnitude. Hence $\rho$ value will be considered in the section 8.4.2 to build the smooth curve.

## 8.3 Planners

A path planner determines a set of points in the free-collision Euclidean space. In that way, two path planning algorithms are introduced as follow.

### 8.3.1 Rapidly-exploring Random Trees RRT

This algorithm builds an incremental computational tree (Pemmaraju y Skiena 2003) based on random sampling. The process is splited into two fundamental stages. First, a random point with a bigger distance than the selected metric is generated. Second, a new point located between the closest point to the random point and the same random point is assigned. (see Algorithm 7).

**Algorithm 7** RRT Algorithm

**Require:** define params
**Require:** $G_i(V).add \leftarrow p_{init}$;
**Require:** $G_i(V).add \leftarrow p_{Goal}$;
**while** $!(q_{rand} = q_{goal})$ **do**
  $q_{rand} = rand(x, y, z)$;
  **if** $!(q_{rand}.collision)$ **then**
    $q_{near} \leftarrow findNearest.neighbor(G(V), q_{rand})$;
    $q_{new} \leftarrow planner.connect(G(V), q_{near})$;
    $G(V).add \leftarrow q_{new}$;
    $G(E).add \leftarrow planner.connect(q_{new}, q_{near}).edges$;
  **else**
    $q_{rand}.discarded$;
  **end if**
**end while**

The open tree structure algorithm ends when $q_{new} = q_{goal}$. The number of iterations is specified as a termination constraint.

### 8.3.2 Modified Adaptive Cell Decomposition MACD

An adaptive *3-Dimensional* decomposition means to split the environment into a finite set of parts with the same shape but in different sizes. The Modified adaptive cell decomposition (MACD) (Samaniego, Sanchis, Garcia-Nieto y col. 2017) contains a finite set of joint or disjoint voxels (minimum processable unit in three-dimensional matrix shape) Borgefors 1986 $[k_1, k_2, ..., k_n]$ where $k_i$ and $k_j$ are adjacent whenever they share boundaries. Voxels can be stated as: $FREE$ whether there is no occupied space around its boundaries (i.e. $k_i \bigcap obj = 0$), $FULL$ whether its limits are completely occupied by an object (i.e. $k_i \subseteq obj$) and $MIX$, if there is a partial occupation (i.e. $k_i \bigcap obj \wedge k_i \nsubseteq obj = 1$). Algorithm 8 shows this voxel labeling.

**Algorithm 8** MACD Algorithm

**Require:** $n, env.zise(x, y, z)$;
**Require:** $obs = (x_{ObsInf}, x_{ObsUpp}, y_{ObsInf}, y_{ObsUpp}, z_{ObsInf}, z_{ObsUpp})$;
$vx = obs_{x_{Inf}} : env.size(x)/2^n : obs_{x_{Up}}$;
$vy = obs_{y_{Inf}} : env.size(y)/2^n : obs_{y_{Up}}$;
$vz = obs_{z_{Inf}} : env.size(z)/2^n : obs_{z_{Up}}$;
$rectangloid.name = name$;
$rectangloid.boundary = env.size(x, y, z)$;
**for** $i = 1 : n$ **do**
  $rows = rectangloid.size$;
  **for** $nextRow : rows$ **do**
    **if** $rectangloid(nextRow).occup == mix$ **then**
      $boundary = boundaryOctree(rectangloid(nextRow).boundary)$;
      **for** $k = 1 : boundary$ **do**
        $rectangloid(newRow).name$;
        $rectangloid(newRow).boundary$;
        **if** $rectangloid(newRow).boundary(vx, vy, vz).obstacle == obstacle.mix$ **then**
          $rectangloid(newRow).occup == mix$;
        **end if**
        **if** $rectangloid(newRow).boundary(vx, vy, vz).obstacle == obstacle.free$ **then**
          $rectangloid(newRow).occup == free$;
        **end if**
      **end for**
    **end if**
  **end for**
**end for**

Notice that every voxel that has $C_{full}$ properties has not been stored. The goal is to find collision-free spaces $C_{free}$ or $C_{mix}$ which can be sub-divided until the predefined fixed value $n$. Hence, $2^n$ indicates the number of the computational tree structure levels. On the other hand, the smallest voxel size is equal to $(x, y, z)/2^n$, thereby $(x, y, z)$ are the total *3-Dimensional* environment size. Computing this hierarchical representation, complex and more compact, the number of neighbors for each voxel can be determined depending on its size, having a neighborhood of at least three voxels.

## 8.4 Curves definition

The introduced techniques returns a set of vertex and nodes in their result, defined as *Control Points* $P$. Hence we can resolve this set of three-dimensional points as a piece-wise polynomial. Therefore, the $3D$ UAV path $P$ is a set of different vertex $v_i$ with a specific order, which can be expressed as a discrete interpolation sequence.

$$P = v_i = f(t_i) \rightarrow \mathbb{R} \tag{8.7}$$

where $f(t_i)$ is a set of three-dimensional points and $i = 1, \ldots, n$ ($n$ is the total number of nodes) is a set of known nodes from the path planning. Therefore, a set of $k$ sub-intervals between $i = 1$ and $i = n$ partitioned in $[a, b]$ can be defined

$$\begin{aligned} [a,b] = [t_1,t_2] \cup [t_2,t_3] \cup \ldots \cup [t_{n-2},t_{n-1}] \cup [t_{n-1},t_n] \\ a = t_1 \leq t_2 \leq \ldots \leq t_{n-1} \leq t_n = b \end{aligned} \tag{8.8}$$

Hence, it exists $s(t)$ with $n$ piece-wise polynomials as

$$s(t) = \begin{cases} s_1(t), & t \in [t_0, t_1] \\ s_2(t), & t \in [t_1, t_2] \\ \quad \vdots & \\ s_n(t), & t \in [t_{n-1}, t_n] \end{cases}$$

$$\begin{aligned} s_j(t) = q_k t^k + q_{k-1} t^{k-1} + \cdots + q_1 t^1 + q_0 t^0 \\ j = 1, 2, \ldots, n \end{aligned} \tag{8.9}$$

where $q_k$, are constant coefficients, $k$ is the degree of polynomial $s_j(t)$. Then $s(t)$ is a spline interpolation (Isaac J Schoenberg 1988; Isaac Jacob Schoenberg 1946) function of degree $k$ for the discrete sequence $P = v_i = f(t_i)$.

In polynomial interpolation, a common effect is the well-known Runge's oscillation phenomenon (Runge 1901). The functions for interpolation by splines curves (Isaac J Schoenberg 1988) minimize the roughness subjected to restrictions, in addition to their extrapolation in several dimensions.

### *8.4.1 De Boor's algorithm*

De Boor's algorithm (Yamaguchi 2012) offers a numerical stability for evaluating spline curves $s(x)$ at point $x$. De Boor is built from a sum of B-spline functions $B_{i,p}(x)$ multiplied by the control vector point $P_i$. B-splines of order $p+1$ are connected piece-wise polynomial functions of order $p$ defined over a nodes grid $t_0, \ldots, t_i, \ldots, t_m$. On the other hand, De Boor's algorithm uses $O(p^2) + O(p)$ operations to evaluate the spline curve in B-spline form.

$$s(x) = \sum_i c_i B_{i,p}(x) \tag{8.10}$$

B-splines polynomials are positive in a finite domain and zero elsewhere.

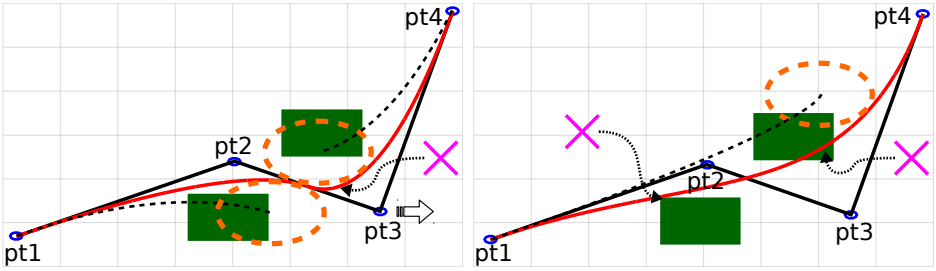
$$B_{i,0} := \begin{cases} 1 & if\ t_i \leq x \leq t_{i+1} \\ 0 & otherwise \end{cases} \tag{8.11}$$

$$\begin{aligned} B_{i,p}(x) = & \frac{x - t_i}{t_{i+p} - t_i} B_{i,p-1}(x) + \\ & \frac{t_{i+p+1} - x}{t_{i+p+1} - t_i} B_{i+1,p-1}(x) \end{aligned} \tag{8.12}$$

The algorithm does not compute the B-spline functions $B_{i,p}(x)$ directly. Instead, it evaluates $s(x)$ through a iterative equivalent equation. Define as $d_{i,r}$ control points with $d_{i,0}$ for $i = k-p, \ldots, k$. For $r = 1, \ldots, p$ the following equation is applied

$$\begin{aligned} d_{i,r} = (1 - \alpha_{i,r}) d_{i-1,r-1} + \alpha_{i,r} d_{i,r-1}; \\ i = k - p + r, \ldots, k \\ \alpha_{i,r} = \frac{x - t_i}{t_{i+1+p-r} - t_i} \end{aligned} \tag{8.13}$$

The iterations are complete when $s(x) = d_{k,p}$, meaning $d_{k,p}$ is the desired result. This allows evaluating different degrees of splines.




(**a**)Splines $2^{nd}$ order (**b**)Splines $3^{rd}$ order

**Figura 8.3:** Spline curve built with 4 control points.

The Fig. 8.3 shows an execution example of splines with different order, the number of control points is 4 (blue circumferences), the track starts from $p_1$ to $p_4$, the final trajectory is the red line and green boxes are obstacles. The splines calculate the smooth path from the *control points*, however, the curvature radio constraint (eq. 8.6) is not considered.

Therefore, the Fig. 8.3(a) shows a $2^{nd}$ order spline where the curvature is out of $\rho$ boundaries (magenta cross). This problem can be solved by moving the *control points*, nevertheless, this action is not collision free. In (Kwangjin Yang y Sukkarieh 2008) the problem is solved by addings *control points* in critical areas (new *control points*) between $(p_1, p_2)$, $(p_2, p_3)$ and $(p_3, p_4)$.

On the other hand, the Fig. 8.3(b) shows an example of $3^{rd}$ order spline curve where the $\rho$ limits are satisfy, nevertheless, the trajectory changes and 2 collisions are produced (magenta cross).

Hence, the better solution is to modify the curvature rather than the control points but change the curvature in critical points. Therefore, the new approach considers the UAV constraints and propose a *3-Dimensional* variation curve with circumference base, detailed below.

### *8.4.2 Semi-circumference curvature approach*

Whether it is considered the number of *control points* $P$ equal to $nP$, exists a set of equations of a Line equal to $nL = nP - 1$ and the set of angles between the equation Lines is $n\theta = nL - 1$ then, each equation Line $rL$ is determined as

$$rL_{i=2}^{nL} = \begin{pmatrix} x & y & 1 \\ P_{(i)}x & P_{(i)}y & 1 \\ P_{(i-1)}x & P_{(i-1)}y & 1 \end{pmatrix} \tag{8.14}$$

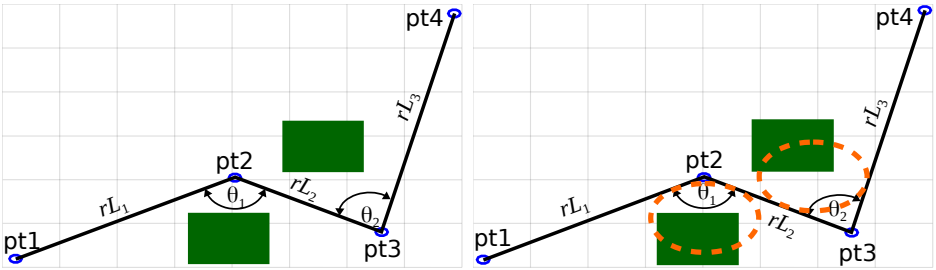
and the set of angles $\theta$ is defined as

$$\begin{aligned} \tan\theta_{i=2}^{n_\theta} &= \left| \frac{m_{(i)} - m_{(i-1)}}{1 + m_{(i)} \cdot m_{(i-1)}} \right| \\ m_{i=2}^{n\theta} &= \frac{P_{(i)}y - P_{(i-1)}y}{P_{(i)}x - P_{(i-1)}x} \end{aligned} \tag{8.15}$$

The main goal is to find the way to locate a circumference $cE$

$$cE_{i=1}^{nL-1} = (x-a)^2 + (y-b)^2 - r^2 \tag{8.16}$$

with a *curvature radio* $\rho$ between two Lines, maintaining the *control points* from the path planning and adding a smooth curve to the critical points shown in Fig. 8.4.




(**a**)Angle between two Lines.

(**b**)Approach circumference intersecting 2 lines.

**Figura 8.4:** Semi circumference curve approach with 4 control points.

The methodology is described above and shown in the Fig. 8.5.

1) *Locate 2 relevant points.-* One point $P_A$ in direction $rL_1$ to distance of curvature radio $\rho$, another point $P_B$ in direction $rL_2$ at the same distance of curvature radio $\rho$ is located.

2) *Perpendicular Lines.-* A perpendicular Line $R_A \perp rL_1$ is build in the point $P_A$ and another Line $R_B \perp rL_2$ because of $P_B$.

3) *Intersection location.-* The Lines $R_A$ and $R_B$ intersect in some place of the coordinates axis, point denoted as $e$.

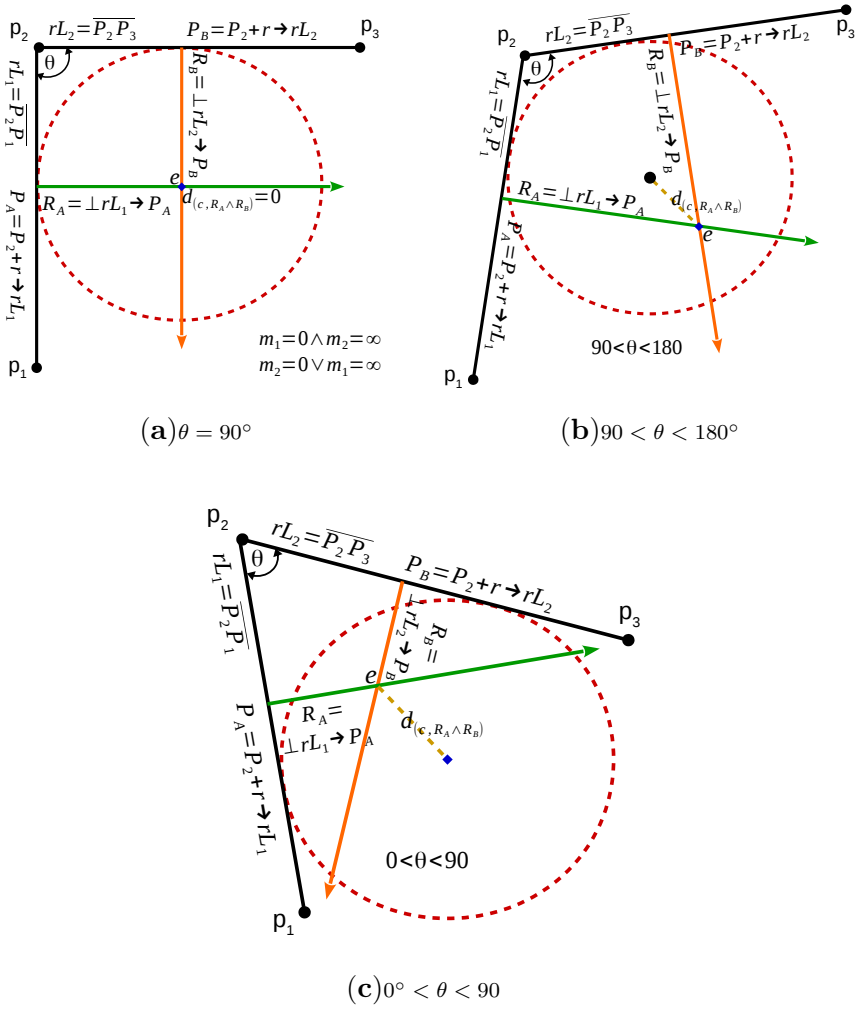
4) *Determine the centre of the circumference.-* To complete the proposed process, 3 different scenarios have been considered to define the central point.

   a) The ideal case is when the angle between $rL_1$ and $rL_2$ is 90°, in that way the intersection point between $R_A$ and $R_B$ determines the new circumference center shown in Fig. 8.5(a). A new intersection between this circumference equation and the Lines $rL_1$ and $rL_2$ determines the new partial track with athe ppropriated curvature.

   b) This case is shown in Fig. 8.5(b). Whether the angle $\theta$ formed between $rL_1$ and $rL_2$ increase. The distance $d_{(c,R_A \wedge R_B)}$ to the intersection $R_A \wedge R_B$ moves away describing an exponential curve as:

$$f(x) = \frac{(p_1 \cdot x^2 + p_2 \cdot x + p_3)}{(x^3 + q_1 \cdot x^2 + q_2 \cdot x + q_3)} \tag{8.17}$$

   c) In the same way of the case $B$, exist an angle $\theta$ between $rL_1$ and $rL_2$ which decrease, this action provokes an estrangement of the circumference center to the intersection between $R_A$ and $R_B$. This distance change is exponential and it has been resolved with the equation 8.17.




(**a**)$\theta = 90°$

(**b**)$90 < \theta < 180°$

(**c**)$0° < \theta < 90$

**Figura 8.5:** Semi circumference curve approach.

At this moment, the new circumference has been defined. Next step is to determine the intersection points between the Lines $rL_1$, $rL_2$ and the circumference. The defined Line (eq. 8.14) can be expressed as a vector $\overrightarrow{rL_{i=2}^{nL}} = \overrightarrow{d} + \gamma\widehat{n}$ where $d$ is the position vector of a point to a Line, $\widehat{n}$ is a unit vector in the line direction, and $\lambda$ is a parameter which slides by the line boundaries. On the other hand, a vector circumference since

eq. 8.16 as $(\overrightarrow{cE_{i=1}^{nL-1}} - \overrightarrow{c})^2 = r^2$, where $\overrightarrow{c}$ is the position vector of the circumference center, and $r$ is the radius. The intersection of the both systems plugs in as:

$$\left(\overrightarrow{d} + \gamma\widehat{n} - c\right)^2 = r^2 \tag{8.18}$$

Therefore, the possible solutions are defined in

$$\begin{aligned} \overrightarrow{x}_{\pm} &= \overrightarrow{d} + \left(A \pm \sqrt{A^2 - B}\right) \cdot \widehat{n} \\ A &= \left(\overrightarrow{c} - \overrightarrow{d}\right) \cdot \widehat{n} \\ B &= \left(d^2 + c^2 - r^2 - 2\overrightarrow{d} \cdot \overrightarrow{c}\right) \end{aligned} \tag{8.19}$$

where $B$ determines the possible intersection points between the lines $(rL_1, rL_2)$ and the new circumference $cE$.

The complete process describes the curve in one plane $(x, y)$. However, the curve contained in plane $(x, z)$ performs the identical described process with unique condition of identical length on the plane $(x, y)$.

## 8.5 Experiments ans Results

The Table 8.1 describe the start *3-Dimensional* environment conditions, where *cInit* is the UAV start point and *cGoal* is the arrival point. The complete environment size in meters is $[1180, 2789, 300]$. On the other hand, the altitude parameters are relevant to complete the *3-Dimensional* navigation.

**Tabla 8.1:** UAV constraints

| **UAV constraints** | | |
|---|---|---|
| ***cInit*** | [39.42070241, -0.42318433, 270] | ° grad. dec. |
| ***cGoal*** | [39.44575112, -0.43378172, 180] | ° grad. dec. |
| ***uavDimensions*** | [2.4, 17.45, 0.42] | m |
| ***curvature radius*** | 33 | m |
| ***flight speed*** | 18.16 | m/s |
| ***pitch*** | ± 21 | ° grad. |
| ***roll*** | ± 21 | ° grad. |

The described environment has been modified, locating obstacles in different places. The Table 8.2 shows the specified location for each obstacle. The column *obs* shows the number of obstacles in the scene, the column *Location* shows the geodesic location of each obstacle and the *Dimensions* column shows the obstacles dimensional characteristics.

**Tabla 8.2:** Obstacles characteristics and locations

| *scene* | *obs* | *Location* | | | **dimensions** |
|---|---|---|---|---|---|
| | | ***lat.* °*dec*** | ***lon.* °*dec*** | ***alt. m*** | **(x, y, z) m** |
| *1* | *1* | 39.433226765 | -0.428483025 | 0 | [13, 26, 30] |
| *2* | *1* | 39.436226765 | -0.426983025 | 150 | [11, 22, 35] |
| | *2* | 39.430226765 | -0.431483025 | 150 | [22, 22, 30] |
| *3* | *1* | 39.435926765 | -0.425783025 | 100 | [6, 6, 20] |
| | *2* | 39.429326765 | -0.432383025 | 200 | [6, 6, 20] |
| | *3* | 39.428726765 | -0.426683025 | 130 | [13, 13, 30] |
| | *4* | 39.433226765 | -0.428483025 | 120 | [20, 20, 60] |

### *8.5.1 Path planning*

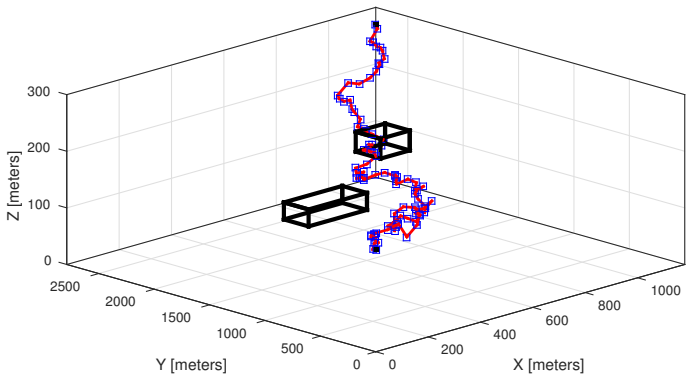
Hence, with the described environment several tests were accomplished. The results begin with the planners evaluation (in section 8.3), 6 test have been executed for each scene. The statistics of those accomplished results are shown in the Table 8.3. The column *Iterations* shows the number of times for execution in the loop with *RRT* until converging to a result, in *MACD* is the number of times the approached decomposition was built. *Distance* column evaluated in kilometers, shows the average distance in direct flight crossing every one of the *control points*.

**Tabla 8.3:** Planners Results

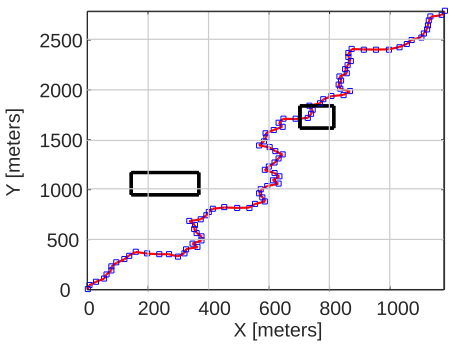
| ***Planner*** | ***scene*** | ***# Iterations*** | ***Distance km*** | ***# Vertex*** | ***# Control Points*** |
|---|---|---|---|---|---|
| ***RRT*** | *1* | 260 | 3.663 | 250 | 89 |
| | *2* | 368 | 3.684 | 350 | 90 |
| | *3* | 192 | 3.585 | 189 | 87 |
| ***MACD*** | *1* | 1016 | 3.369 | 378 | 19 |
| | *2* | 616 | 3.259 | 336 | 13 |
| | *3* | 808 | 3.298 | 414 | 21 |

Such as can be seen in the above Table, the difference between the stochastic algorithm *RRT* and the discrete algorithm *MACD* in the column *Control Points* is relevant, the difference is around 5 times in stochastic algorithm. In the column *Vertex*, the discrete algorithm is 1,2 times bigger than stochastic.

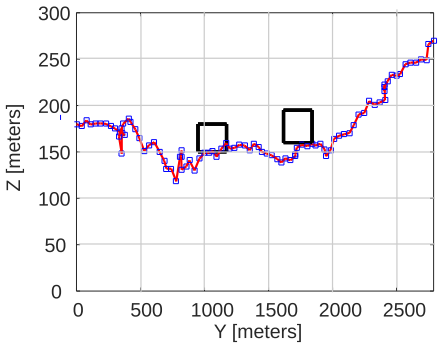
Nevertheless, a visual perspective of the aforementioned Table 8.3 can be appreciated in Fig. 8.6, where it is shown the graphical results in the execution of *scene 2* with *RRT* algorithm. This algorithm has been configured as bi-directional and the value is dynamic, in that way the distance generated is a nearby value to 42. The curvature radius of this algorithm is bigger to ensure a path generation. The Fig. 8.6(a) shows a *3-Dimensional* vision while the Fig. 8.6(b) and Fig. 8.6(c) shows the $(x, y)$ and $(y, z)$ axis respectively.




**(a)** 3D view $(-13, 42)$



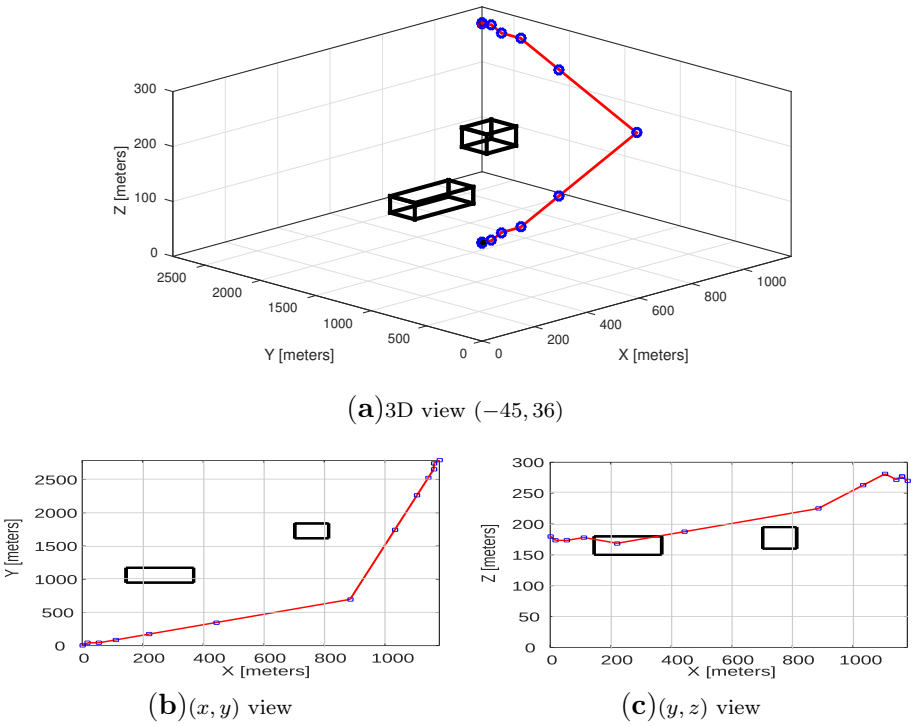

**(b)** $(x, y)$ view




**(c)** $(y, z)$ view

**Figura 8.6:** Resulting path planning of RRT.

On the other hand, the Fig. 8.7 shows the results in the same *scene 2* where the Fig. 8.7(a) shows a $3D$ view, the lateral views are shown in Fig. 8.7(b) and Fig. 8.7(c). As the fixed-wing UAV has limited mobility, besides stopping the flight is not possible, the best option of the path is *MACD*. In that way, this set of results have been used to evaluate the curves analyzed in section 8.4.



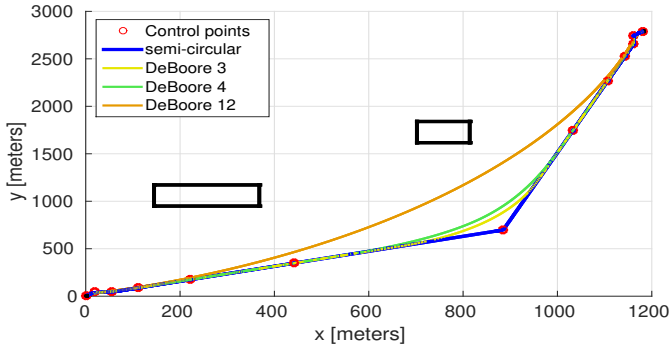

**(a)** 3D view $(-45, 36)$

**(b)** $(x, y)$ view

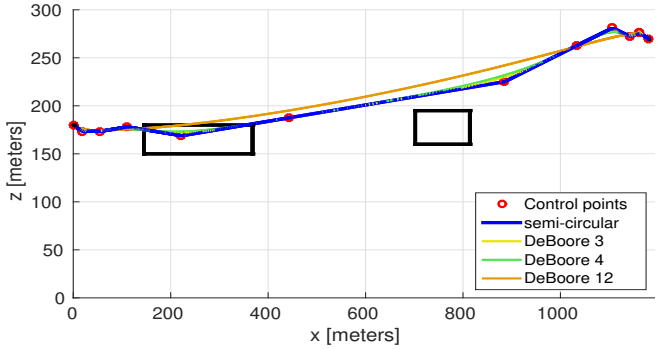
**(c)** $(y, z)$ view

**Figura 8.7:** Resulting path planning of MACD.

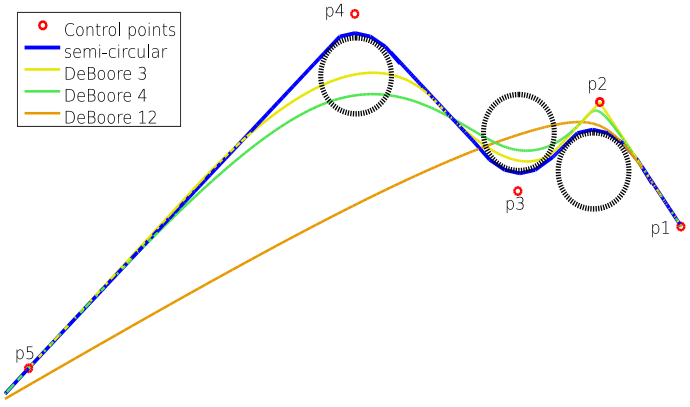
Assuming constant velocities and accelerations, the curvature variation is the resulting angle from the calculation of *cAct* and *cNext*, where *cAct* is the UAV actual theoretical point in the curve and *cNext* is determined by the point in the curve where the distance is equal or higher than the minimal predefined distance *vertex*. DeBoore algorithm has been applied to evaluate the splines of different orders and compared against the semi-circular curve in section 8.4.2.




(**a**)Final curves, view $(x, y)$.




(**b**)Final curves, view $(x, z)$.




(**c**)Final curves, zoom in view $(x, z)$.

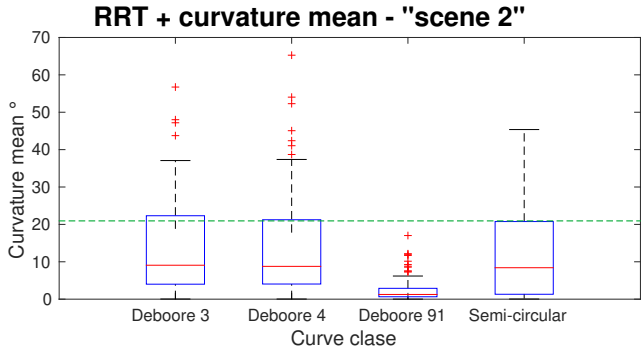
**Figura 8.8:** Final curves from control points of *MACD* in *scene 2*.

The Fig. 8.8 shows the final curves build by the aforementioned techniques. Fig. 8.8(a) and Fig. 8.8(b) shows the lateral views $(x, y)$ and $(x, z)$. The red circumferences are the control points, the blue line is the semi-circular curve, yellow, green and orange lines are DeBoore with a different order. On the other hand, the Fig. 8.8(c) shows a zoom in the upper right side of the figure 8.8(b), and visualizes the control points $p1$, $p2$, $p3$, $p4$, $p5$. The dotted black circumference shows the maximum curvature for the UAV. In that way, where *DeBoore 3* and *DeBoore 4* generate an approach to $p2$ surpassing the UAV maneuverability capacity.

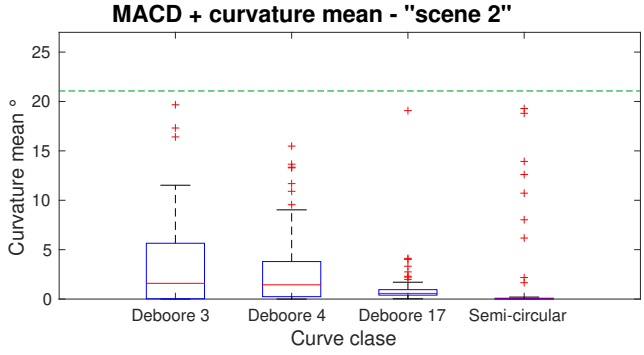
A curvature statistics produced by the curves of *scene 2* is depicted in the Fig. 8.9. The number of *control points* generated by the scene, built with *RRT* is 92. Hence, each smooth planning algorithm *DeBoore and Semi-Circular curve* returns an equal number of *3-Dimensional* points, the curve length for this example is 12117 points.

The Fig. 8.9(a) shows the results in curvature terms (°), applying DeBoore algorithm of order $3^{rd}$, $4^{th}$ and $91^{st}$ order and *semi-circular curve* on the path generated by *RRT* Fig. 8.9(a). Nonetheless, in *DeBoore 3* and *DeBoore 4* the UAV angle pitch ($\pm 21°$, see Table 8.1) constraint was exceeded 32 times and 25 times respectively (over the green line in Fig. 8.9). *DeBoore 91* and *semi-circular curve* never exceedes the UAV pitch constraint, but *DeBoore 91* is very close to the obstacles.

The Fig. 8.9(b) shows the curvature with the same algorithms on the path generated by *MACD*, both cases show similar results, the absolute mean-variance is concentrated in the low values.




(**a**)Curvature, absolute mean-variance on smooth curve with RRT path.

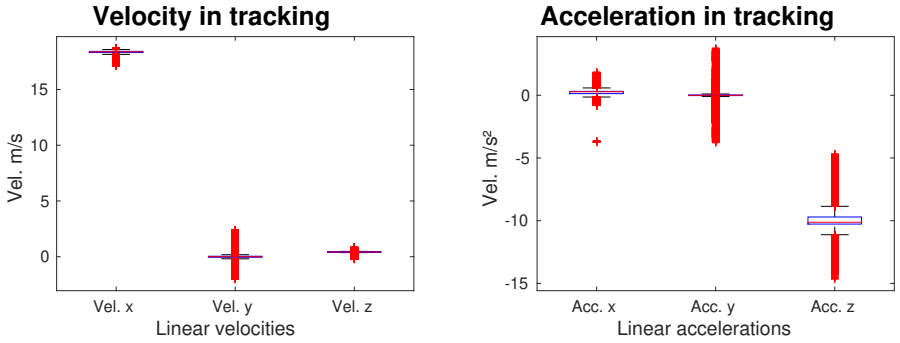



(**b**)Curvature, absolute mean-variance on smooth curve with MACD path.

**Figura 8.9:** Ratio of absolute mean variation curvature.

In order to verify the effectiveness of the described methodologies and the theoretical results, an environment simulation based on Matlab-Simulink is used, which interface with the flight simulator software *FlightGear* (Perry 2004).

On the other hand, during the flight, it has been recollected 186121 samples of the flight dynamic, the results in velocities and accelerations variations after finishing the smooth trajectory are shown in Fig. 8.10 and Fig. 8.10(a). In the same way, the linear accelerations are shown in Fig. 8.10(b).




(**a**)Mean of angular velocities. (**b**)Mean of angular accelerations.

**Figura 8.10:** Ratio UAV dynamic in velocities and accelerations in flight.

The velocities and accelerations keep its mean value, that means the maneuvers executed by UAV are within its constraint range. For instance in *Vel. x* the advance velocity, the mean is close to 18, the UAV velocity hold continuous flight. The different velocities and accelerations shown are produced by the twist in the curve.

## 8.6 Conclusions

In this paper, a comparative study of motion planning for fixed-wing UAVs in 3-Dimensional space has been discussed. The main contributions have been the inclusion of constraints and boundaries in the path generation (based on the kinematic and dynamic UAV model), and the use of stochastic data to drive the algorithms.

In the path planning field, a discrete environment reconstruction offers a direct way to arrive from *init* to *goal* points. Stochastic data gives a response less direct in direction terms but, if the control points produced by the stochastic methods have the appropriate distance the UAV might track the final smooth path.

In summary, smooth path curve generation adding UAV maneuverability constraints may be a good approach to ensuring traceability, constant velocities and a continuous flight. In addition, these results could increase the flight time of small UAVs.

Finally, in future work, new tests should be carried out on Hardware-In-the-Loop to ensure a secure implementation on real UAV platforms as the *kadett 2400*.

## Capítulo 9

# Smooth 3D Path Planning by Means of Multiobjective Optimization for Fixed-Wing UAVs

**Nota sobre el capítulo:** La generación de trayectorias suaves es un proceso geométrico y puede causar un costo computacional muy elevado, por lo que se ha planteado una solución en base a una optimización multiobjetivo. El contenido de este capítulo aparece en el siguiente trabajo:

> Samaniego, F., Sanchis, J., García-Nieto, S., & Simarro, R. (2020). *Smooth 3D Path Planning by Means of Multiobjective Optimization for Fixed-Wing UAVs*. Electronics, 9(1), 51., Impact Factor: 1.764, JCR: Q2, SJR: Q1, DOI:https://doi:10.3390/electronics9010051.

***Abstract:*** *Demand for 3D planning and guidance algorithms is increasing due, in part, to the increase in unmanned vehicle-based applications. Traditionally, two-dimensional (2D) trajectory planning algorithms address the problem by using the approach of maintaining a constant altitude. Addressing the problem of path planning in a three-dimensional (3D) space implies more complex scenarios where maintaining altitude is not a valid approach. The work presented here implements an architecture for the generation of 3D flight paths for fixed-wing unmanned aerial vehicles (UAVs). The aim is to determine the feasible flight path by minimizing the turning effort, starting from a set of control points in 3D space, including the initial and final point. The trajectory generated takes into account the rotation and elevation constraints of the UAV. From the defined control points and the movement constraints of the UAV, a path is generated that combines the union of the control points by means of a set of rectilinear segments and spherical curves. However, this design methodology means that the problem does not have a single solution; in other words, there are infinite solutions for the generation of the final path. For this reason, a multiobjective optimization problem (MOP) is proposed with the aim of independently maximizing each of the turning radii of the path. Finally, to produce a complete results visualization of the MOP and the final 3D trajectory, the architecture was implemented in a simulation with Matlab/Simulink/flightGear.*

## 9.1 Introduction

The latest technological and scientific advances in the field of mobile robotics have enabled the area of autonomous vehicles (AV) to become a reality applied to the civil and military sectors (Fagnant y Kockelman 2015; Kyriakidis, Happee y Winter 2015). Consequently, this technological branch presents a constant and vertiginous development in different fields, among them the development of new navigation and guidance techniques. The constant evolution in this field responds to new challenges in real applications (Münzer y col. 2006; Y. Morales y col. 2013; Krotkov y Hebert 1995).

Autonomous vehicles with non-holonomic characteristics (Rodríguez-Seda 2014; Ren y Z. Cai 2010; Dixon y Frew 2007) can be technologically adapted to dif-

ferent environments, and so, produce unmanned ground vehicles (UGVs) (Jun, Saut y Benamar 2016; B. Li y col. 2016; Mekonnen, S. Kumar y Pathak 2016), unmanned underwater vehicles (UUVs) (J. Xu, M. Wang y L. Qiao 2015; Gafurov y Klochkov 2015; Qi y Z.-j. Cai 2018), and unmanned aerial vehicles (UAVs) (Ramasamy y col. 2016; Zhu, Xianghong Cheng y Yuan 2016; Chee y Zhong 2013; Courbon y col. 2010). Obviously, each of the above categories presents its own scientific and technological challenges, including path planning, navigation, and guidance.

It is important to note that the most common problem when determining a possible feasible path is the consideration of the AV intrinsic constraints. Therefore, the non-inclusion of kinematic and/or dynamic constraints of the AV when addressing the path planning problem may lead to non-feasible solutions that, for example, make it impossible for the AV to satisfactorily follow a path. However, including in the design all the constraints of the AV in the calculation phase of the path planning can lead to very complex optimization problems without a single solution and with very high computational costs.

This paper focuses on the generation of smooth paths for fixed wing unmanned aerial vehicles (UAVs) that have high flight capabilities and extended flight-time missions, even in low power propulsion situations (Hull 2007). For these reasons, fixed-wing UAVs are suitable for use in terrain mapping applications for later action by the security forces, and in search and rescue tasks, both for the detection of people and the provision of first aid.

Due to the non-holonomic constraints of fixed-wing UAVs, the aim is to create a smooth three-dimensional curve from an initial point to a goal point through a complex 3D space with or without obstacles. To achieve this goal, it is essential to define a feasible path that minimizes flight turning effort and distance traveled. The ordered set of waypoints that will be used to generate the path to follow is defined as the control points set. In general, the problem of path planning is defined in a space region denoted as $\chi$ and split as the tripla $(\chi_{free}, \chi_{init}, \chi_{goal})$. The movement space is defined as $(\mathbb{W} = \mathbb{R}^N : N = \{2, 3\}$, where the obstacle region is denoted by $\chi_{obs}$, so that $\chi/\chi_{obs}$ is an open set denoting the collision-free space $\chi_{free}$. The initial condition $\chi_{init}$ and the final condition $\chi_{goal}$ are elements of $\chi_{free}$.

The set of control points that define the collision-free space is calculated using specific path planning methods based on continuous and discrete environment sampling. Some examples of these techniques are: the rapidly-exploring random tree (RRT) (Abbadi y col. 2012b; W. Aguilar y S. Morales 2016; W. G. Aguilar y col. 2017; Yao, Honglun Wang y Su 2015); probabilistic road maps

(PRM) (Yan, Y.-S. Liu y Xiao 2013; Yeh y col. 2012; Denny y Nancy M. Amatoo 2013b; Q. Li y col. 2014b; Ortiz-Arroyo 2015b); heuristic planners (genetic algorithms—GA) (Xianyi Cheng y C. Yang 2008; Liang y L. Xu 2009); swarm intelligence (J. Liu y col. 2017; Cao y col. 2016; Duan y P. Qiao 2014; Alhanjouri y Alfarra 2011); fuzzy logic (Bakdi y col. 2017; Pandey y Parhi 2017); Voronoi diagrams (Thanou y Tzes 2014; Y. Qu, Yintao Zhang y Youmin Zhang 2014; Fang, Luan y Z. Sun 2017); artificial potential (Khuswendi, Hindersah y Adiprawita 2011; Xia Chen y J. Zhang 2013; Rivera, Prieto y Ramirez 2012; Lifen y col. 2016); and recursive rewarding modified adaptive cell decomposition (RR-MACD) (Samaniego, Sanchis, Garcia-Nieto y col. 2018).

All the techniques mentioned build piece-wise paths in 2D or 3D to address the standard problem of path planning. These methods may provide optimal or near-optimal paths; however, they cannot guarantee smoothness and continuity, which could make it difficult to guide the UAV through the paths generated. Moreover, these techniques do not directly incorporate the operational constraints of the UAV and the environment. Therefore, this paper proposes a methodology to define feasible and smooth UAV paths, including system operational kinematic constraints.

The ability of an UAV to fly from one position to another and the consequent definition of the mission to be performed remains a challenge that requires the application of increasingly sophisticated strategies. One of the fundamental constraints to be considered in mission planning is the ability to be positionable and sensorially oriented (on the UAV or the environment) throughout the duration of the mission. This sensory location enables the construction of maps of the environment, and also enables the UAV to estimate its own current position and complete its self-location on the map. Similarly, it is important to note that good positioning and sensory orientation can be achieved by making a path plan and tracking it across or beyond the sensory detection domain. It is important to mention that the definition of smooth paths is not a new subject; various approaches have been proposed for non-holonomic UAVs, such as Dubins (Dubins 1957; Beard y McLain 2012) in which paths are defined by connecting lines and arc-circular segments. The disadvantage is the generation of discontinuities in the connection points between segments. Another useful methodology in the literature is Clothoid curves (Fleury y col. 1995; Vanegas y col. 2018; Brezak y Petrović 2013), the main advantage being increases in curvature as a function of the arc-length; meanwhile the disadvantage lies in the limitation of length. These approximate curves generate continuities (B. Barsky y T. DeRose 1989) of the type $C^1$ (a continuous path $C^1$ preserves the continuity of the tangent vector, in addition to maintaining continuous speed)

and have been used in applications on mobile vehicles and on UAVs with flight limitation at a constant altitude, and examples are mentioned in (Kim y col. 2014; Isaacs y Hespanha 2013; Masehian y Kakahaji 2014; Fraichard y Scheuer 2004; Pepy, Lambert y Mounier 2006). An intuitive approach that ensures $C^2$ continuity (a continuous trajectory $C^2$ preserves the second order differential values at each point of the trajectory, in addition to maintaining the continuity of the acceleration vector) in 3D planning focuses on curvature and torsion zero at the junction points, as proposed in (Girbés, Vanegas y Armesto 2019). This is an interesting proposal where the 3D curve is built from a 2D curve in a $(x, y)$ plane; the resulting length of this curve is the main parameter to build the curve in the other $(x, z)$ plane. Finally, the Bézier, B-spline (De Lorenzis, Wriggers y Hughes 2014; Pigounakis, Sapidis y D 1996) are easy to implement polynomial curves; however, none are suitable for planning since they are sensitive to control parameters and weights (Pérez y col. 2018), and do not take into account the constraints of the vehicles on which they are applied, so those curves need to be optimized. Finally, these types of parametric curves have no physical meaning and the relationship between design parameters and system variables is not defined. The above-mentioned approximation methodologies generate two different phenomena called interpolation (generation of a curve which must pass through the control points) and approximation (generation of a curve that approximates the control points, but may only go through the start and finish rather than all of them) detailed in (Huh y S.-R. Chang 2014; S.-R. Chang y Huh 2014). The work presented here explores both phenomena in-depth to respond to the definition of feasible trajectories.

The starting point of this work is the set of 3D collision-free points generated by the various planners that guarantee that the path departs from the init point and ends at the goal point, and avoids static and dynamic obstacles. From these collision-free points, an ordered set of straight lines is built that define a first path that will later be smoothed to incorporate the feasibility and constraints of the UAV. The UAV constraints in this work are focused on its ability to turn horizontally and vertically. Therefore, for a UAV to complete a sequence of turns at a defined speed, it must determine its minimum turning radius $Rp$. If the turning radius is too small, the UAV will lose the trajectory; however, if the turning radius grows, the UAV can perform maneuvers with less effort.

The aim is to maintain 3D planning results, and at the same time, generate a finite set of possible 3D curves that optimize an approximate 3D curve, and simultaneously, the turns of the UAV—which is raised as a multiobjective optimization problem (MOP) (Herrero y col. 2007). This approach will result

in a set of paths that meet UAV constraints expressed as dominant solutions on a Pareto $n$ dimensional front (Yaochu Jin y Sendhoff 2008). Finally, selection criteria must be applied to determine the desired response from the point of view of curvature $\kappa$ and torsion $\tau$ of the generated 3D curve. Thus, in order to verify the functionality of the proposed methodology, the results of the curves generated after the 3D curve optimization were compared with a known Bézier type approximation methodology (G. Farin 2014).

This document is structured as follows. A brief summary of MOP concepts is given in Section 9.2.1. In Section 9.2.2, a brief description of smooth curves is given. Section 9.3 presents the formulation of the problem. Section 9.4 details the complete methodology for solving the problem. Section 9.5 details the experiments and results of 3D smooth path planning. Finally, conclusions and future work are presented in Section 9.6.

## 9.2 Background

### *9.2.1 MultiObjetive Optimization*

The optimization problem (OP) attempts to determine a solution that represents the optimal value (minimum or maximum) of a function, such as $f : X \to \mathbb{R}$, where $X$ is a feasible decision vector, being mín$(f(x)) : x \in X$. However, for problems where simultaneous optimization of more than one objective is necessary, i.e., multiobjective optimization (MOP), the function is shaped $f : x \to \mathbb{R}^k$, where $k \geq 2$ is the number of objectives. Therefore, the value vector of the target function could be defined as $f : X \to \mathbb{R}^k, f(x) = (f_1(x), \ldots, f_k(x))^T$. However, there is not usually a single $X$ that generates an optimum that simultaneously satisfies each of the $k$ objectives, due to the conflict between the objectives. The aim is to find a situation in which all objectives are satisfactorily within acceptable parameters. The MOP solution leads to points where any improvement in one target results in the degradation of any other target (one or more). Thus, these points are represented as a Pareto front (Yaochu Jin y Sendhoff 2008), where all the points of the front are equally optimal.

Therefore, as expressed in (Herrero y col. 2007) the MOP can be established as

$$\text{mín } \mathbf{J}(\theta) = \min_{\theta \in D}[J_1(\theta), J_2(\theta), \ldots, J_m(\theta)] \tag{9.1}$$

subject to:

$$\begin{aligned} g(\theta) &\leq 0 \\ h(\theta) &= 0 \\ \theta_{il} \leq \theta_i \leq \theta_{iu}, i &= [1, \ldots, n], \end{aligned} \tag{9.2}$$

where $\theta \in \mathbb{R}^n$ is the decision vector, $D$ is the decision space; $\mathbf{J}(\theta) \in \mathbb{R}^m$ is the target vector; $g(\theta)$ and $h(\theta)$ are constraint vectors; and finally, $\theta_{il}$ is the upper boundary and $\theta_{iu}$ is the lower boundary of the decision space. Consequently, there is no single optimal model; in fact, there is a set of optimal solutions with different trade-offs between objectives, where none is better than the others. Using the definition of dominance, the Pareto set $\Theta_P$ is the set of each non-dominated solution.

In this way, the Pareto domination is defined in case a solution $\theta^1$ dominates another solution $\theta^2$; that is, $(\theta^1 \prec \theta^2)$, if

$$\forall i \in B, J_i(\theta^1) \leq J_i(\theta^2) \wedge \exists k \in B : J_k(\theta^1) < J_k(\theta^2), \tag{9.3}$$

where $J_i(\theta)$, $i \in B := [1 \ldots m]$ are the objectives to be optimized. Therefore, the optimal set of Pareto $\Theta_P$ is given by

$$\begin{aligned} \Theta_P = \theta \in D | \nexists \tilde{\theta} \in D : \tilde{\theta} \prec \theta \\ J(\Theta_p) = \{J(\theta) | \theta \in \Theta_p\}, \end{aligned} \tag{9.4}$$

where $\Theta_p$ and $J(\Theta_p)$ are MOP solutions. However, in most cases they are unreachable because $\Theta_P$ normally includes infinite solutions. Therefore, a finite set of $\Theta_P^*$ from $\Theta_P$ and another finite set of $J(\Theta_p^*)$ from $J(\Theta_p)$ represent satisfactory solutions. Starting from $J(\Theta_p^*)$, the decision maker selects a solution according to the established preferences. For example, a certain point in the Pareto front that is close to the ideal point (called utopia point) $\mathbf{J}^{ideal}$.

$$\mathbf{J}^{ideal} = \{J_{1\ min}(\theta), \ldots, J_{m\ min}(\theta)\}. \tag{9.5}$$

Hence, an appropriate methodology for characterizing MOP is known as the elitist multiobjective evolutionary algorithm ($\not\nearrow$–$MOGA$) (Herrero y col. 2007), which makes a distributed approach to Pareto's front. The aim of $\not\nearrow$–$MOGA$ is to find an intelligent distributed convergence towards a set of $\epsilon$-Pareto; i.e., determine $\theta_{P\epsilon}^*$ along the Pareto front $\mathbf{J}(\Theta_P)$. The target space is split into a fixed number of *boxes*. Therefore, for each dimension $i \in B$, cells $n\_box_i$ wide are created $\epsilon_i$, where

$$\epsilon_i = \left(J_i^{max} - J_i^{min}\right)/n\_box_i$$
$$J_i^{max} = \max_{\theta \in \Theta_{P\epsilon}^*} J_i(\theta),\ J_i^{min} = \min_{\theta \in \Theta_{P\epsilon}^*} J_i(\theta). \tag{9.6}$$

Each *box* can be occupied by a single solution; therefore, this grid produces an intelligent distribution and preserves the diversity of $\mathbf{J}(\Theta_{P\epsilon}^*)$. In addition, it is important to note that only the occupied boxes are verified, avoiding the need to use other clustering techniques to obtain adequate distributions. On the other hand, for a solution $\theta \in D$, $box_i(\theta)$ is defined as

$$box_i(\theta) = \left[\frac{J_i(\theta) - J_i^{min}}{J_i^{max} - J_i^{min}} \cdot n\_box_i\right] \forall_i \in B \tag{9.7}$$

Then, $\mathbf{box}(\theta) = \{box_1(\theta), \ldots, box_m(\theta)\}$. A solution $\theta^1$ with value $\mathbf{J}(\theta^1)$ $\epsilon$ dominates the solution $\theta^2$ with value $J(\theta^2)$, denoted by $\theta^1 \prec_\epsilon \theta^2$, only if

$$\mathbf{box}(\theta^1) \prec \mathbf{box}(\theta^2) \vee \left(\mathbf{box}(\theta^1) = \mathbf{box}(\theta^2)\ and\ \theta^1 \prec \theta^2\right). \tag{9.8}$$

So, a set $\Theta_{P\epsilon}^* \subseteq \Theta_P$ is $\epsilon$-stop only if

$$\forall \theta^1, \theta^2 \in \Theta_{P\epsilon}^*, \theta^1 \neq \theta^2, \mathbf{box}(\theta^1) \neq \mathbf{box}(\theta^2)\ and\ \mathbf{box}(\theta^1) \not\prec_\epsilon \mathbf{box}(\theta^2). \tag{9.9}$$

Thus, a $\Theta_{P\epsilon}^*$ is reached with the greatest possible number of solutions that adequately characterize Pareto's front and whose number of possible solutions depend on $n\_box_i$, and will not exceed the following level.

$$|\Theta_{P\epsilon}^*| \leq \frac{\Pi_{i=1}^n n\_box_i + 1}{n\_box_{max} + 1}, n\_box_{max} = \max_i n\_box_i. \tag{9.10}$$

Hence, it is possible to control the maximum number of solutions to characterize the Pareto front. Finally, due to the definition of the box, the anchor points $J_i(\theta^{i*})$ are assigned a value of $box_i(\theta^{i*}) = 0$, whereby $J_i(\theta^{i*}) = J_i^{min}$. Therefore, no solution $\theta$ can $\epsilon$-dominate them because, by applying the definition of the box, their $box_i(\theta) \geq 1$.

The above process delivers two defined sets of responses: (a) the Pareto front $\Theta_P^*$ deploys a finite set of minimum values as the optimal path response within the search space; (b) the corresponding optimal points $J(\Theta_j^*)$.

### 9.2.2 3D Curves for UAVs

A non-holonomic UAV (Velasco-Carrau y col. 2016) can perform flights in 3D Euclidean space. Nevertheless, to complete each movement sequence (horizontal and vertical), a set of UAV flight constraints must overcome. An UAV attempts to perform 3D movements at a defined velocity, meaning that the UAV is moving continuously, attempting to maintain that velocity. However, an UAV has a maximum capacity of turn and elevation at a defined velocity. Therefore, the aim is to build a 3D smooth curve inside the UAV flight turning boundaries, in such a way to reach a complete 3D smooth curve tracking.

A smooth curve can be defined as the representation of a continuous function, which can be expressed as $\boldsymbol{C} : I \to X$, where $I$ is the curve interval composed of real numbers, while $X$ represents the topological space. If the topological space is three-dimensional $X = \mathbb{R}^3$, then $\boldsymbol{C} : [a, b] \to \mathbb{R}^3$, is a differentiable injectable and continuous function, whose arc-length $s$ is independent of the parametrization $\boldsymbol{C}$. If a UAV is considered as a particle that travels along the envelope of the $\boldsymbol{C}$ curve at a defined speed $v$, this particle suffers changes in its local coordinate system due to the set of rotations of the $\boldsymbol{C}$ curve. Therefore, $\boldsymbol{C}$ must not allow rotation changes outside of the intrinsic rotation capabilities of the UAV.

It is important to mention that the formulation known as Frenet–Serret (Honig, Schucking y Vishveshwara 1974; Iyer y Vishveshwara 1988) describes the kinematic properties of particles that move along three-dimensional Euclidean space $\mathbb{R}^3$ continuous and differentiable $\boldsymbol{C}(s)$, and parametetrized by its arc-length $s$ (the arc-length is an invariant Euclidean characteristic of the curve). However, if we assume a curve given by a series of points along the Euclidean space as $r(t)$, where the parameter $t$ does not need the arc-length, then the tangent (T), normal (N) and bi-normal (B) derived vectors can be described, as all are mutually perpendicular (orthogonal base). Therefore, according to the theory of differential geometry of curves (Abbena, Salamon y Gray 2017), the following equals can be defined as:

$$\begin{aligned} T(t) = N(t) \times B(t) &= \frac{r'(t)}{\|r'(t)\|} \\ B(t) = T(t) \times N(t) &= \frac{r'(t) \times r''(t)}{\|r'(t) \times r''(t)\|} \\ N(t) = B(t) \times T(t) &= \frac{[r'(t) \times r''(t)] \times r'(t)}{\| [r'(t) \times r''(t)] \times r'(t)\|} \end{aligned} \tag{9.11}$$

where $r'(t) = \frac{dr(t)}{dt}$, $r''(t) = \frac{d^2r(t)}{dr^2}$ and $r'''(t) = \frac{d^3r(t)}{dr^3}$ are the position vector derivatives $r(t)$. These three vectors configure a navigation reference system of the UAV. Similarly, it is important to mention that the tangent vector $T(t)$ is parallel to velocity, while the normal vector $N(t)$ is represented by the change of direction per time unit of velocity.

The curvature terms $\kappa$ (change of direction of the vector tangent $T(t)$ to the curve $r(t)$) and torsion $\tau$ (change of direction of the vector bi-normal $B(t)$) are defined as:

$$\kappa = \frac{\|r'(t) \times r''(t)\|}{\|r'(t)\|^3} \tag{9.12}$$

$$\tau = \frac{r'(t) \cdot [r''(t) \times r'''(t)]}{\|r'(t) \times r''(t)\|^2}. \tag{9.13}$$

where $\kappa$ indicates a direct correlation with the horizontal rotation capability of the UAV, while $\tau$ indicates the elevation capability of the UAV. Therefore, the trihedron Frenet–Serret can be defined in matricial notation as a skew–symmetric matrix:

$$\begin{bmatrix} \dot{T} \\ \dot{N} \\ \dot{B} \end{bmatrix} = \begin{bmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{bmatrix}, \tag{9.14}$$

where the point over the variable indicates the derivative with respect to the parameter of arc-length $s$.

In summary, the space curve according to the formulation Frenet–Serret, defines a smooth curve that will be built from a spatial point and its tangent vector; this curve will be generated in the Euclidean 3D space depending on the pre-defined values of $\kappa$ and $\tau$, and finalize at a spatial 3D point after completing the arc-length $s$. However, our goal was to start from a defined point $p_{init}$, and build a curve that touches a target point $p_{goal}$; therefore, the values of $\kappa$ and $\tau$ have to fit so that when complete, the arc-length $s$ will touch $p_{goal}$. That gives us an infinite number of possible values of $\kappa$ and $\tau$ with which we could meet that goal. Even if the maximum and minimum values of $\kappa$ and $\tau$ are bounded, the complexity of the problem is high. In (Lazard, Reif y Hongyan Wang 1998) it is mentioned that the complexity in a 2D environment becomes NP-hard, and the need is demonstrated for path planning algorithms that generate short paths with bounded curvatures in complicated environments. The aim is to find for possible approximate values of $\kappa$ and $\tau$ that do not

exceed the turn capabilities of the UAV. In other words, starting from Figure 9.1, and assuming a configuration of the UAV as a triplet $(\rho, \kappa, \tau)$, where $\rho = [P_1, \ldots, P_5]$ is a dimensional vector that specifies $n$ collision-free points, $\kappa$ and $\tau$ are a set of curvatures and torsion along the path. The smooth curve starts from $P_1$ and reaches out to $P_5$, and approaches the remaining $\rho$ without affecting them, with values of $\kappa$ and $\tau$ within the boundaries established by the maneuverability capabilities of the UAV. In summary, the selection of $\rho_{goal}$ points which determine radii of the tangent curves to the $\rho$ points, will be obtained by solving a multiobjective optimization problem (MOP). This MOP is stated in such a way that the values of all radii will be maximized simultaneously. Obviously, the optimizer handles these values, taking into account that they are in conflict (as radius of one of them is increased, consequently, the adjacent radius are reduced. Therefore, the MOP solver will try to find a trade-off solution that guarantees the best set of points $\rho_{goal}$ for all tangent curves between control points $\rho$.

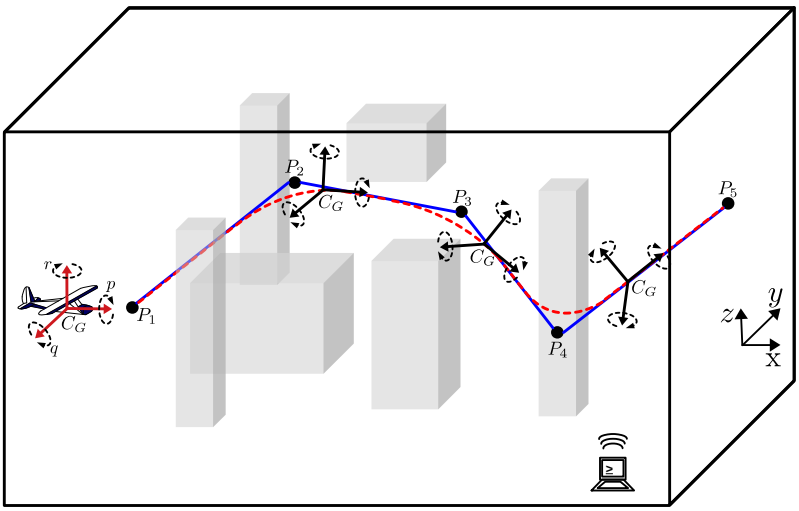



**Figura 9.1:** Perspective of the 3D flight problem for a fixed wing UAV, where $C_G$ represents the position vector of the center of gravity of the UAV. The global coordinate system $CS_g$ is placed at the origin, the orientation of the local body coordinate system $CS_b$ expressed by Euler roll, pitch and yaw angles respectively, which have been defined by three unitary orthogonal vectors aligned with the three axles of the vehicle and centered at $C_G$. Finally, the angular velocities along the local axis of the UAV $X$, $Y$ and $Z$ are represented by $p$, $q$ and $r$, respectively. The set of collision-free points $P_i$ is represented by black dots; the blue line describes the discrete path built from 3D path planning; the red dotted line is the new smooth optimized path that the UAV could follow.

## 9.3 Problem Definition

Let us assume a workspace $\mathbb{W} = \mathbb{R}^3$, where it is possible to define a set of static or dynamic obstacles, such as ground or aerial boxes of different dimensions and locations (see Figure 9.1). The in-flight UAV receives data from its control station regarding environmental conditions and performs the necessary calculations to determine the best smooth 3D trajectory. Relevant data include the set of ordered 3D flight waypoints that are collision-free $\rho = [P_1, \ldots, P_5]$ in the environment. The intrinsic maneuverability capabilities are determined by a $Rp$ turning radius (which determines the vertical and horizontal turning limitations) defined by their flight speed. The aim is to start from $\rho_{init}$ and reach $\rho_{goal}$ in such a way that the UAV approaches the direct trajectory marked by the ordered sequence $\rho$. Therefore, $\rho = P_i(x_i, y_i, z_i)$ where $(i = 1, \ldots, n)$ and $n$ is the total set of collision-free spaces and can be expressed as a discrete interpolation sequence $\rho = f(t_i) \rightarrow \mathbb{R}$, where $f(t_i)$ is a set of nodes in 3D space. Therefore, it is possible to establish a set of sub-intervals $(n-1)$ between $i = 1$ and $i = n$ partitioned in $[a, b]$, defined as:

$$\begin{aligned} [a,b] = [t_1,t_2] \cup [t_2,t_3] \cup \cdots \cup [t_{n-2},t_{n-1}] \cup [t_{n-1},t_n] \\ a = t_1 \leq t_2 \leq \cdots \leq t_{n-1} \leq t_n = b. \end{aligned} \tag{9.15}$$

A linear union between pairs of points then results in $\boldsymbol{L} : [a, b] \rightarrow (x, y, z)$. and can be expressed as a set of straight lines that mark a direct flight path $\boldsymbol{L}(t)$ split into $(n-1)$ piece-wise.

$$\begin{aligned} \boldsymbol{L}(t) = \begin{cases} L_1(t) : \ t \in [t_1, t_2] \\ L_2(t) : \ t \in [t_2, t_3] \\ \ \ \vdots \\ L_n(t) : \ t \in [t_{n-1}, t_n] \end{cases} \\ \boldsymbol{L}(t) = L_1(t) + L_2(t) + \cdots + L_n(t). \end{aligned} \tag{9.16}$$

Therefore, $\boldsymbol{L}(t)$ is a linear interpolation function for the discrete sequence $\rho = f(t_i)$. In the same way, between the $\rho$ points, there is a subset of $(n-1)$ straight lines that join the init and the goal of the trajectory along the collision-free flight space.

However, a non-holonomic UAV cannot perform every type of maneuver defined by $\boldsymbol{L}(t)$. In general, it is desirable to perform maneuvers with a high tur-

ning radius. Therefore, the approach presented in this work builds a smooth trajectory from $\rho$, that attempts to avoid inappropriate maneuvers using low values of $\kappa$ and $\tau$ included within the boundaries of the UAV flight turn, while simultaneously closing in on the trajectory $\boldsymbol{L}(t)$.

Let us assume, from Figure 9.1, that the blue line denoted as $\boldsymbol{L}(t)$ is the direct path between the collision-free points of the environment, and the red dotted line is a smooth 3D spatial curve defined as $\boldsymbol{C}(t)$. The construction of this 3D smooth curve $\boldsymbol{C}(t)$ is done by joining a set of segments that can be of two types: spherical curves $S$ (defined from a sphere of radius $Rp$) or straight lines $L$. Thus, each $S$ segment is defined by three continuous points of $\rho$; this segment $S$ has two points of tangency, one for each pair of adjacent straight lines $\boldsymbol{L}(t)$ formed by the current set of three points $\rho$. Hence, each $S$ segment may have infinite solutions, with each radius $Rp$ resulting in different tangent points on the lines $\boldsymbol{L}(t)$. Therefore, for each $S$ segment, an infinite set of spheres can be defined, which will be linked through the relevant $L$ segments or another $S$ segment. Obviously, this approach to the problem suggests the existence of infinite combinations for the $S$ and $L$ segments. The way to address this issue has been through the approach of an MOP.

## 9.4 Methodology

This section describes the proposed methodology for the generation of 3D smooth trajectories. The proposed method is split into two parts, first detailing how the $S$ segments were obtained, and then describing the union with the $L$ segments.

### *9.4.1 Definition of Spherical Segment*

Let us assume that from the result of a path planning, $\rho = [P_1, \ldots, P_n]$ is the set of collision-free points of the environment described in Figure 9.2 (red points). This set of points is defined as $P_i(x_i, y_i, z_i) : i = \{1, \ldots, n\}$, where $p_{init} = P_i(x_i, y_i, z_i) : i = \{1\}$ and $p_{goal} = P_i(x_i, y_i, z_i) : i = \{n\}$.

As indicated above, Figure 9.2(b) shows an osculating sphere ($oS$) Abbena, Salamon y Gray 2017 defined with a minimum turning radius value $Rp$, located between the set of the first 3 $P_i$ and tangential to the straight lines $\boldsymbol{L}(t)$ formed between the same set of $P_i$. Therefore, taking into account the number of collision-free points $\rho$, the set of spheres is equal to $G_i : i = \{1, \ldots, n-2\}$, as shown in Figure 9.2(c) (orthogonal view).

Figure 9.2(b) shows the first $G_i : i = 1$ located among the three first $P_i$s. Therefore, it is possible to define a plane $\pi_i : i = 1$ between the same points $P_i$, which will have an angle in relation to the location of the current set of points $P_i$, as can be seen in the Figure 9.3(a), 9.3(b). The importance of the definition of this plane is given by the fact that within it is contained the center of $G_i$ with radius $Rp$. In this way, there is a self-contained curve $S_i$ (as a series of points along the Euclidean space) on the surface of the sphere and tangent to $\boldsymbol{L}(t)$ with $t_2$ and $t_3$ in plane $\pi_i$, as shown in Figure 9.3; hence, the $S_i$ curve segment (black line) is defined as:

$$\begin{array}{l} S_i(t) = [S_x, S_y, S_z] \\ \left.\begin{array}{ll} S_{i_x} & = x_0 + Rp * \sin(\psi) * cos(\varphi) \\ S_{i_y} & = y_0 + Rp * \sin(\psi) * sin(\varphi) \\ S_{i_z} & = z_0 + Rp * \cos(\psi) \end{array}\right\} \begin{array}{c} \varphi_1 \geq \varphi \geq \varphi_2 \\ \wedge \\ \psi_1 \leq \psi \leq \psi_2. \end{array} \end{array} \tag{9.17}$$

where, $x_0$, $y_0$ and $z_0$ together represent the center of $G_i$. The curve $S_i$ performs a horizontal and vertical path due to the angular ranges of $\psi$ and $\varphi$, which implies variations in the values of $\kappa$ and $\tau$ (these have a direct connection to $Rp$ and the arc-length of $S_i$). Consequently, if the value of $Rp$ grows, $S_i$ also grows, while $\kappa$ and $\tau$ decrease.

**Remark 9.4.1** *If the plane $\pi_i$ is parallel to the horizontal plane $(x, y)$ of the environment, then $\tau = 0$ because the movements of the UAV will be horizontal. In the same way, if $\pi_i$ is parallel to the vertical plane $(x, z)$ of the environment, then $\kappa = 0$.*

However, before applying Equation (9.17), it is necessary to determine the location of the points $(x_0, y_0, z_0)$ so that $G_i$ is tangent at a point on its surface with $\boldsymbol{L}(t)$, as shown in Figure 9.2(b) on the points ($t_2$ and $t_3$). Nevertheless, it should be noted that there is an angle between each pair of $\boldsymbol{L}(t)$, and this leads to $G_i$ approaching or moving away from the lines and their tangent points. Therefore, the geometric analysis applied to arrive at an optimal solution is detailed below.

First, a vector direction in space can be defined as $\overrightarrow{v} = p - q : p \wedge q \in \mathbb{R}^3$. Therefore, starting from the known data $\rho = [P_1, \ldots, P_n]$, taking Figure 9.2 as an example, where it is assumed that the collision-free initial points are $(P_i : \{i = 1, \ldots, 3\})$, a first set of two vectors is defined as:

$$\left.\begin{array}{l} \overrightarrow{u}_i = p - q : p = P_{(i+1)}, q = P_{(i)} \\ \overrightarrow{v}_i = p - q : p = P_{(i+1)}, q = P_{(i+2)} \end{array}\right\}, i = 1. \tag{9.18}$$

Just like a perpendicular vector from $\overrightarrow{u}_i$ to $\overrightarrow{v}_i$, denoted as $\overrightarrow{\eta}$, the normal vector is defined as:

$$\overrightarrow{\eta} = \overrightarrow{u}_i \times \overrightarrow{v}_i. \tag{9.19}$$

Consequently, the parametric equation of the $\pi_i$ plane containing three points is defined as:

$$\pi_i = \begin{bmatrix} (x - p_x) \\ (y - p_y) \\ (z - p_z) \end{bmatrix} * [\overrightarrow{\eta}] \Bigg\} : p_x = P_{(i+1)_x}, p_y = P_{(i+1)_y}, p_z = P_{(i+1)_z}. \tag{9.20}$$

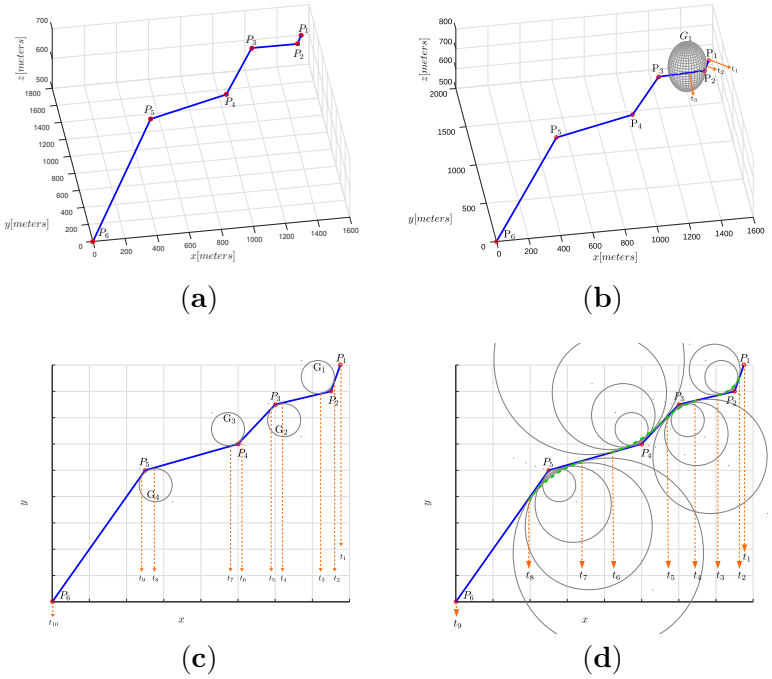



**Figura 9.2:** Smooth path planning problem. The red dots represent $\rho$ and the blue line is the path made up of straight-lines $\mathbf{L}(t)$. (**a**) Result of path planning with $\rho$ collision-free points. (**b**) Definition of a sphere with a relation of the $Rp$ minimum. (**c**) Set of $G_i$ over $\rho$. (**d**) Example of optimal smooth trajectory, with optimized $\kappa$ and $\tau$ represented by dotted green lines in orthogonal view.

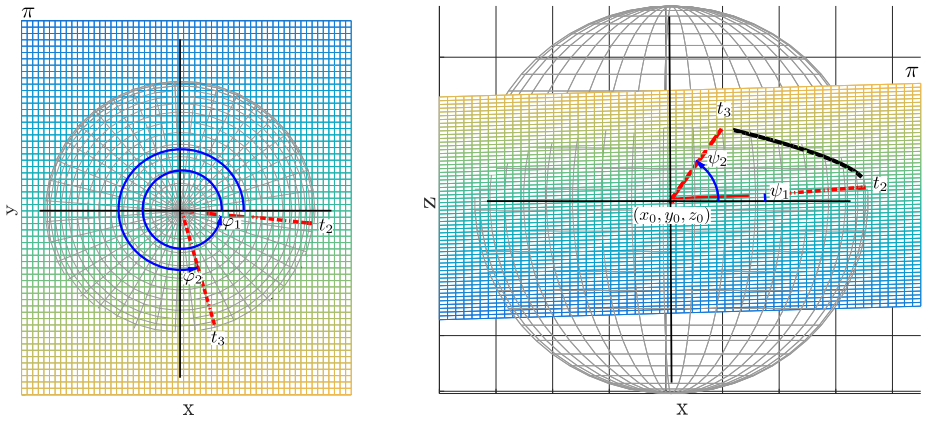



**Figura 9.3:** Spherical smooth path, where the black line shows the curve in a segment $S_i$ along the plane $\pi_i$. The red lines show the union from the center of coordinates of the sphere to the points of intersection between the sphere and the $\pi_i$ plane resulting in the spherical semicurve. (**a**) View perpendicular to the horizontal plane $(x, y)$. (**b**) Perpendicular view to the vertical plane $(x, z)$.

In the same way, the Euclidean distance defined between two points $p$ and $q$ is given by

$$d(p,q) = \sqrt{\sum (p-q)^2}. \tag{9.21}$$

Therefore, two distances can be defined as $du_j : p = P_{(i+1)}, q = P_{(i)}$ and $dv_j : p = P_{(i+1)}, q = P_{(i+2)}$. Finally, the angle between $\overrightarrow{v}_i$ and $\overrightarrow{u}_i$ is defined by:

$$\angle(\overrightarrow{u}_i, \overrightarrow{v}_i) = \phi_i = \tan^{-1} \frac{||\overrightarrow{u}_i \times \overrightarrow{v}_i||}{\overrightarrow{u}_i \cdot \overrightarrow{v}_i}. \tag{9.22}$$

Therefore, with Equation (9.22) and $Rp$ known, the tangential points at the lines $\boldsymbol{L}(t)$ can be located at a distance defined as:

$$\sigma_i = \frac{Rp}{\phi_i/2}. \tag{9.23}$$

Thereby, two spatial points defined as $pUi_i$ and $pUg_i$ located in the direction of the vector $\overrightarrow{u}_i$ provide that $P_i = P_{i+1}$, $P_g = P_i$ and $d(p,q) = du_i$; thus

$$\begin{aligned} \gamma &= \sigma_i / d(p,q) \\ pUi_j &= (P_i - P_g) * \gamma + P_i \\ pUg_j &= -(P_i - P_g) * \gamma + P_i. \end{aligned} \tag{9.24}$$

In the same way, two points $pVi_i$ and $pVg_i$ can be defined in the vector direction $\overrightarrow{v}_i$, so long as $P_i = P_{i+1}$, $Pg = P_{i+2}$ and $d(p,q) = dv_i$, according to Equation (9.24). Hence, the perpendicular bisector of $pUi_i$ and $pVi_i$ on the plane $\pi_i$ determines the center of the sphere $(x_0, y_0, z_0)$. Figure 9.4(a) shows the application of the Equations (9.18)–(9.24), which can be repeated throughout the successive collision-free points $\rho$ (this first Algorithm 9 is summarized in pseudocode). Between the centers of the spheres $(x_0, y_0, z_0)$ and the points of intersection with $\boldsymbol{L}(t)$ are the displacement angles $\varphi$ and $\psi$ of the segment $S_i$, as can be seen in the Figure 9.3, where $pUi_i = t_2$ and $pVi_i = t_3$.

**Remark 9.4.2** *Regardless of the angle condition produced by the pair of straight lines $\boldsymbol{L}(t)$ denoted in the Equation (9.22), the angle formed between the points of intersection on the sphere $G_i$, seen from its center towards the vertical or horizontal component, does not exceed $90^o$ in any case; that is, $(0^o < \varphi < 90^o)$ and $(0^o < \psi < 90^o)$.*

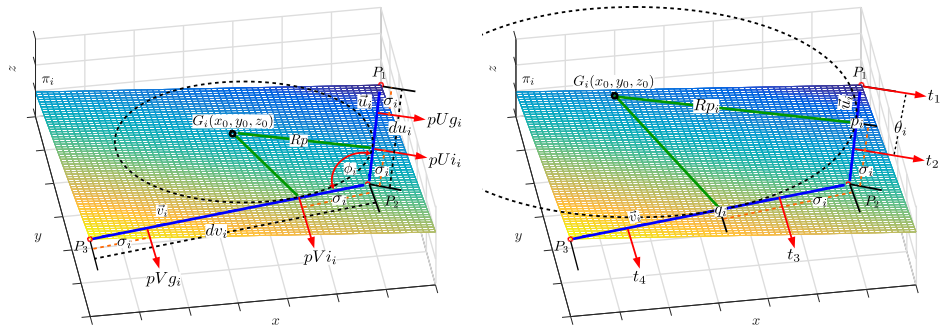



**Figura 9.4:** Description of the methodological basis. The circle of black dots shows the osculating sphere ($oS$); the green line is the radius of turn $Rp_i$. (**a**) Sphere location $G_i$ with minimum turning radius. (**b**) Sphere location $G_i$ with upper turning radius displaced within the bounds given by $[t_1, t_2]$ and defined by the value of $\theta_i$.

**Algorithm 9** First set of $G_i$

1: $\rho = [P_1, \ldots, P_n] \rightarrow$ Path planning problem result.
2: $Rp =$ minimum sphere radius.
3: $k = 1$ counter for the set of intervals $t$.
4: **for** $i = 1 : n - 2$ **do**
5: $(\overrightarrow{u}_i, \overrightarrow{v}_i) =$ vectors .directions;
6: $\phi_i =$ angle.betweeLines$(\overrightarrow{u}_i, \overrightarrow{v}_i)$
7: $\sigma_i =$distance.intersectionPoint$(Rp, \phi_i)$;
8: $(pUi_i, pUg_i) =$intersectPoint.$(P_{i+1}, P_i, \sigma_i)$;
9: $(pVi_i, pVg_i) =$intersectPoint.$(P_{i+1}, P_{i+2}, \sigma_i)$;
10: $G_i(x_0, y_0, z_0) =$bisectPoint$(pUi_i, pVi_i, \pi_i)$;
11: **if** i==1 **then**
12: $t_k = P_1$;
13: $t_{k+1} = pUi_i$;
14: **else**
15: $t_k = pUg_j$;
16: $t_{k+1} = pUi_i$;
17: **end if**
18: **if** i==n-2 **then**
19: $t_{k+2} = pVi_i$;
20: $t_{k+3} = P_n$;
21: **end if**
22: $k = k + 2$;
23: **end for**

Algorithm 9 summarizes the geometric procedure followed. In line 4, a loop is started that performs through all the collision-free points marked by $\rho$. From line 5 to line 9, the necessary computations are made to determine the center of $G_i$ (line 10). The definitions of the intervals containing the $S$ and $L$ segments are in lines 12, 13, 15, 16, 19 and 20.

The described process shows the geometric analysis for the location of the set of spheres $G_i$ defined with constant radius $Rp$, as can be seen in Figure 9.2(c). In addition, there is a set of four segments $S$ and another set of five segments $L$, being segments $S$—those comprised by the intervals $[t_2, t_3]$, $[t_4, t_5]$, $[t_6, t_7]$ and $[t_8, t_9]$, and the segment $L$ included by the intervals $[t_1, t_2]$, $[t_3, t_4]$, $[t_5, t_6]$, $[t_7, t_8]$ and $[t_9, t_{10}]$. The goal now is to increase the radius $Rp$ in each segment, so that the values of $\kappa$ and $\tau$ along the curve are minimized, and the solution is to increase the radius $Rp_i$ in each $G_i$.

The solution adopted in this work is to move the intersection point of each sphere $G_i$ in the direction of the adjacent segment $\boldsymbol{L}(t)$. Consequently, $G_i : i = 1$ approximates symmetrically to the intervals $t_1$ and $t_4$, $G_i : i = 2$ makes the corresponding approximation to the intervals $t_3$ and $t_6$, etc. Therefore, in Figure 9.4(b), the segments $\boldsymbol{L}(t)$ can be seen adjacently to $G_i : i = 1$, denoted as $[t_1 \equiv P_1, t_2 \equiv pUi_j, t_3 \equiv pVi_j, t_4 \equiv pVg_j]$. Thus, between the adjacent intervals $[t_1, t_2]$ a vector is defined $\overrightarrow{u}_i = t_2 - t_1$, and associated with this vector is a spatial point $p_i$ defined by the parametric equation:

$$\left.\begin{array}{l} p_{i_x} = t_{2_x} + \theta_i * \overrightarrow{u}_{i_x} \\ p_{i_y} = t_{2_y} + \theta_i * \overrightarrow{u}_{i_y} \\ p_{i_z} = t_{2_z} + \theta_i * \overrightarrow{u}_{i_z} \end{array}\right\}, 0 \leq \theta_i \leq 1, \tag{9.25}$$

where $\theta_i$ defines the space point $p_i$ along $\overrightarrow{u_i}$ and within the intervals $[t_1, t_2]$. Therefore, the value of the distance $\sigma_i$ from $P_{i+1}$ to $p_i$ is defined according to the equation on (9.21), being $p = P_{i+1}$ and $q = p_{i+2}$. A symbolic space point is defined $q_i$ between the intervals $[t_3, t_4]$ with direction $\overrightarrow{v_i} = t_3 - t_4$ at the same distance $\sigma_i$. Then $\sigma_i$ also has the angle $\phi_i$, and according to the Equation (9.22) it is possible to define a new $Rp_i$ according to Equation (9.23), which would have a higher radius value. Finally, the perpendicular bisector of $p_i$ and $q_i$ on the plane $\pi_i$ determines the center of $G_i(x_0, y_0, z_0)$ (see Figure 9.4(b)). Therefore, as the center of $G_i$ is defined, the Equation (9.17) defines the segment $S_i$—and over this segment we find lower values of $\kappa$ and $\tau$ according to the Equations (9.12) and (9.13).

*Multiobjective Problem Definition (MOP)*

Given Equation (9.25), it is important to note that any value of $\theta_i$ between 0 and 1, defines a space point between the interval $[t_1, t_2]$. In the same way, it is important to mention that within the boundaries of $\theta_i$, there is an infinite number of space points with an infinite number of radius $Rp_i$ and its corresponding infinite number of $G_i$, with which its corresponding segments $S_i$, can be built.

Therefore, to obtain an optimal solution, a multiobjective problem (MOP) is solved using evolutionary algorithms (Coello Coello, Pulido y Montes 2005) based on the concept of $\epsilon$-dominance (Laumanns y col. 2002). To do this, it is necessary to define the decision variables, the initial conditions of the process, the constraints of the MOP and the index vector to be optimized to represent the Pareto front. If it is assumed that the number of spheres $G_i$ is equal to $m$, and the number of objectives for each $G_i$ is equal to two, then

$J^{ideal}(\theta) = [J_1(\theta), J_2(\theta), \ldots, J_{2*m}(\theta)]$ is the objectives vector, where $J_i$ denotes the $i^{th}$ objective. Consequently, $J_i^A = \text{mín}(\kappa(\theta_i))$, $J_i^B = \text{mín}(\tau(\theta_i)) \in G_i : [i = 1, \ldots, m]$, where $J_i^A$ and $J_i^B$ depend on the decision variables vector $\theta$. Assume $D$ as a decision space within a subset $\mathbb{R}^D$, where $\theta$ is the decision variable vector composed of a set of $\theta_i$ for all $i \in 1 < i < m$, where $\theta_i$ is $[0, 1]^D$. Consequently, the MOP problem can be stated as:

$$\min_{\theta \in D} [J_i^A(\theta), J_i^B(\theta)]_{1\times(2*m)}, \ \forall i \in 1 \leq i \leq m. \tag{9.26}$$

$$\text{where: } J_i^A = \frac{\|S_i'(t) \times S_i''(t)\|}{\|S_i'(t)\|^3}, \text{ from eq. (9,12)}$$

$$J_i^B = \frac{S_i'(t) \cdot [S_i''(t) \times S_i'''(t)]}{\|S_i'(t) \times S_i''(t)\|^2}, \text{ from eq. (9,13)}$$

$$\theta = [\theta_i]_{1\times m}, \ \forall i \in 1 \leq i \leq m$$

subject to:

$$S_i(t) = \begin{cases} S_{i_x} = & x_0 + Rp_i * \sin(\psi) * cos(\varphi) \\ S_{i_y} = & y_0 + Rp_i * \sin(\psi) * sin(\varphi) \\ S_{i_z} = & z_0 + Rp_i * \cos(\psi) \end{cases}, \text{ from eq. (9,17)}$$

$$Rp_i = \sigma_i * (\phi_i/2), \text{ from eq. (9,23)}$$

$$\sigma_i = \sqrt{\sum (p_i - P_{i+1})^2}, \text{ from eq. (9,21)}$$

$$p_i = \begin{cases} p_{i_x} = t_{2_x} + \theta_i * \overrightarrow{u}_{i_x} \\ p_{i_y} = t_{2_y} + \theta_i * \overrightarrow{u}_{i_y} \\ p_{i_z} = t_{2_z} + \theta_i * \overrightarrow{u}_{i_z} \end{cases}, \text{ from eq. (9,25)}$$

$$\theta_i \in [0, 1]^D.$$

In summary, the aim is to find an optimal 3D smooth curve that minimizes $\kappa$ and $\tau$ in each of the possible $S_i$. It is important to mention that the adjacent spheres $G_i$ can grow into each other, until a maximum of $q_i \in \overrightarrow{v_i} \equiv p_{i+1} \in \overrightarrow{u_i}$, which implies a decrease in the total set of segments, as described Figure 9.2(d), where the green dotted line shows the set of $S_i$ segments belonging to $G_i$.

An example of reconstruction according to the response $\Theta_P^*$ can be seen in Figure 9.2(d), where the $S$ reconstruction is made in four segments, defined by the boundaries $[t_2, t_3]$, $[t_4, t_5]$, $[t_5, t_6]$ and $[t_7, t_8]$; the $S$ segments belonging to $\boldsymbol{C}(t)$ are defined according to Equation (9.17).

In contrast, and with reference to Figure 9.2(d), the $L$ segments are defined by the rest of the boundaries, those boundaries being $[t_1, t_2]$, $[t_3, t_4]$, $[t_6, t_7]$ and $[t_8, t_9]$.

### 9.4.2 Definition of Straight-Line Segment

A segment path of $L$ in a straight-line can be described by two points in the Euclidean space. Figure 9.2(d) shows an example of an $L$ segment defined by the $[t_1, t_2]$ points, where the direction of the line is given by the flight path of the UAV. Therefore, $\overrightarrow{u}$ (Figure 9.5) is a unit vector that points in the direction of the desired orientation, and with $d$ defined as the distance between $t_1$ and $t_2$ according to Equation (9.21). Therefore, the $L$ segments will be described, in general, as:

$$L(t) = \{r \in \mathbb{R}^3 : r = (t_1 - t_2) * \gamma - t_1\} \rightarrow 0 \leq \gamma \leq d. \tag{9.27}$$

Finally, the interpolation of $S$ and $L$ build a final 3D smooth curve on the plane $(x, y, z)$.

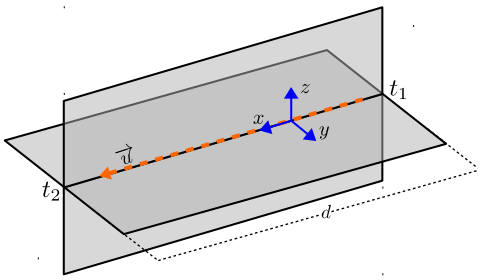



**Figura 9.5:** Straight-line segment.

## 9.5 Experiments and Results

This section presents the results of the computer simulation using Matlab/ Simulink software and flightGear flight simulator for graphical visualization.

In this section we analyze five scenarios in 3D space, similar to the methodology proposed in (Samaniego, Sanchis, García-Nieto y Raúl Simarro 2019). Recursive rewarding modified adaptive cell decomposition (RR-MACD) splits the 3D

environment like a discrete mesh of collision-free voxels. In particular, it places the UAV within an initial voxel and determines which of the adjacent voxels is the most suitable to make a displacement. To determine the best choice of displacement, the set of adjacent voxels have a set of associated constraints (conditions such as distance, vertical, and horizontal movement angles, battery consumption, etc.) to be satisfied before performing a discrete displacement. The voxel that minimizes the total effort and satisfies the constraints will be the next collision-free point. The RR-MACD methodology gives two sets of results based on the defined constraints. The results presented in (Samaniego, Sanchis, García-Nieto y Raúl Simarro 2019) are shown in summary form in Table 9.1, where the first column shows the scenario number. The second column shows RR-MACD with four constraints and the RR-MACD with 10 constraints in the third column shows the conditions to solve the path planning problem. The 3D control points reflected in Table 9.1, $\rho_x(F) \approx \rho$, are the starting points for analyzing the method described in this paper for generating 3D smooth curves. Finally, it is important to note that the algorithms have been run on an Intel(R) Core(TM) i7-4790 3.60 GHz CPU (Manufacturer: Gigabyte Technology Co., Ltd., Model: B85M-D3H) with 8Gb RAM and S.O. Ubuntu Linux 16.04 LTS. The algorithms were programmed in MATLAB version 9.4.0.813654 (R2018a).

**Tabla 9.1:** 3D path planning results. The number of free-collision voxels within a defined environment is indicated as $S_{free}$, while the number of discrete 3D nodes built by (Samaniego, Sanchis, García-Nieto y Raúl Simarro 2019) is denoted by $\rho_x(F)$ (3D final path).

| **Env.** | **RR-MACD 4 Constraints** | | **RR-MACD 10 Constraints** | |
|---|---|---|---|---|
| | **#** $\boldsymbol{S_{free}}$ | **#** $\boldsymbol{\rho_x(F)}$ | **#** $\boldsymbol{S_{free}}$ | **#** $\boldsymbol{\rho_x(F)}$ |
| **# 1** | 115 | 18 | 202 | 27 |
| **# 2** | 27 | 8 | 35 | 10 |
| **# 3** | 19 | 6 | 16 | 7 |
| **# 4** | 11 | 6 | 51 | 10 |
| **# 5** | 19 | 7 | 35 | 10 |

It is important to mention that the characteristics of the UAV assumed in the experiments have been taken from (Velasco-Carrau y col. 2016), whose study has been carried out on a fixed wing UAV model *kadett 2400*, represented by six states $(x, y, z, \phi, \theta, \psi)$, where the first three states define the position vector

of the UAV's global coordinate system, located at the origin of its center of gravity. The last three are the Euler angles of roll, pitch and yaw respectively, which define the orientation of the UAV.

Finally, the aim of the simulations is for the UAV to maintain its continuous flight at a defined constant speed of 18 m/s, within an established minimum radius of curvature $Rp = 33$ m to achieve smooth behavior and without maneuvers that could endanger the integrity of the aircraft.

It is important to remember that because the number of $\rho_x(F) = [P_1, \ldots, P_n]$ collision-free points is greater than five, a proper visualization method is essential to the decision making process for the final solution. Thus, the graphical representation method called level diagram (Blasco y col. 2008) has been used, which consists of representing each objective and each design parameter in separate diagrams, all of which are synchronized with their $y$ axis. Synchronization is made with the normalized distance of each point from the Pareto front to the ideal point. Therefore, with a brief training this representation offers a good visual understanding of the compromise reached on the Pareto front.

### 9.5.1 Bézier

To compare the results, an approximation of curves has been applied using Bézier (Abbena, Salamon y Gray 2017), which enables the generation of trajectories for non-holonomic systems through a set of discrete control points, where the curve in a multidimensional environment can be described as:

$$\begin{aligned} B(t) &= \sum_{i=0}^{n} P_i b_{i,n}(t), \ t \in [0,1] \\ b_{i,n}(t) &= \binom{n}{i} t^i (1-t)^{n-1}, \ i = 0, \ldots, n \end{aligned} \tag{9.28}$$

Bernstein-Bézier generates a finite vector of points belonging to the curve and guarantees to go through the first and last control points (translated as $\rho = [P_1, \ldots, P_n]$) and remaining inside the convex envelope.

### *9.5.2 Application Example*

To represent visual and numerical results from Table 9.1, the results for the #3 environment with RR-MACD with four constraints are detailed below. As in this example, the total number of $P_i$ equals 6, and this means that the decision criteria of Equation (9.26) $\rightarrow m = 4$. Therefore, there are four values of $\kappa$ and four values of $\tau$; i.e., $\Theta_p^* = (J_1(\theta_1) = \kappa_1,\ J_3(\theta_2) = \kappa_2,\ J_5(\theta_3) = \kappa_3,\ J_7(\theta_4) = \kappa_4,\ J_2(\theta_1) = \tau_1,\ J_4(\theta_2) = \tau_2,\ J_6(\theta_3) = \tau_3$ and $J_8(\theta_4) = \tau_4)$, as can be seen in Figure 9.6.

It is important to mention that the interpolation of segments $S$ and $L$ builds a set of 3D smooth curves, all of which are possible solutions. It is, therefore, necessary to address a decision stage (decision maker (DM)) that selects one of them; i.e., a point on the Pareto front. In this work, the selection criteria based on the shortest distance to the ideal point have been used. Figures 9.6 and 9.7 show the selected point in red from $J(\Theta_p^*)$ and $\Theta_p^*$, which have been selected using the $\infty$-norm standard.

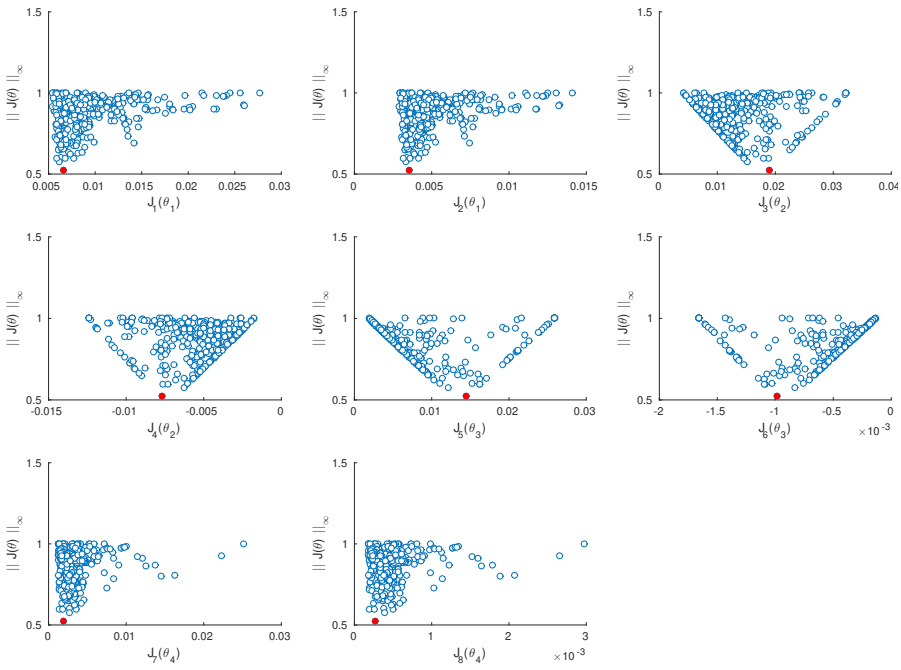



**Figura 9.6:** Representation of Pareto front using $\infty - norm$. The $J$ even sub-indices represent the $\kappa$ values in each $S_i$ segment, while the $J$ odd sub-indices represent the $\tau$ values in the same $S_i$ segments. Targets closest to $J^{ideal}$ are shaded in red circles.

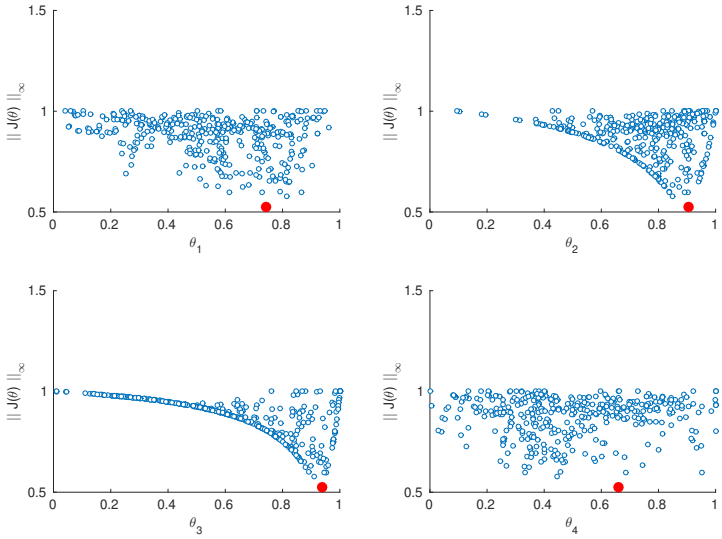



**Figura 9.7:** Representation of Pareto's optimal parameters. Targets closest to $J^{ideal}$ are shaded in red circles.

Figure 9.8(a) shows the build of the 3D smooth curve $\boldsymbol{C}(t)$, while Figure 9.8(b) shows the best optimization in terms of curvature $\kappa$ and total torsion $\tau$, according to Equations (9.12) and (9.13). It is important to note that the mathematical mean of the values of the geometric variables $\kappa$ and $\tau$ tends to be low. However, in some particular cases an increase is detected due to the change in direction of the flight; i.e., when the UAV is terminating an $S$ segment in one direction and another $S$ segment begins in the opposite direction. The curve generated by Bézier $B(t)$ is shown as a yellow line, a more direct route can be seen between the init and goal point. However, this curve is very close to the bottom obstacle. To solve this, different authors propose modifying the control points $\rho$, or adding new points within the points established initially, as remarked in Section 9.1.

Figure 9.9 shows the set of four additional examples from Table 9.1, in order to represent the functionality of the algorithm. Similarly, it is important to note that the number of $\rho$ was different in each experiment, as were the altitudes, which guaranteed movement and 3D planning. It is also important to mention that the first environment shown in Figure 9.9(a) has smaller flight dimensional characteristics, so the turning radius in this example was set at $Rp = 3$ m with an average flight speed of 1.7 m/s.

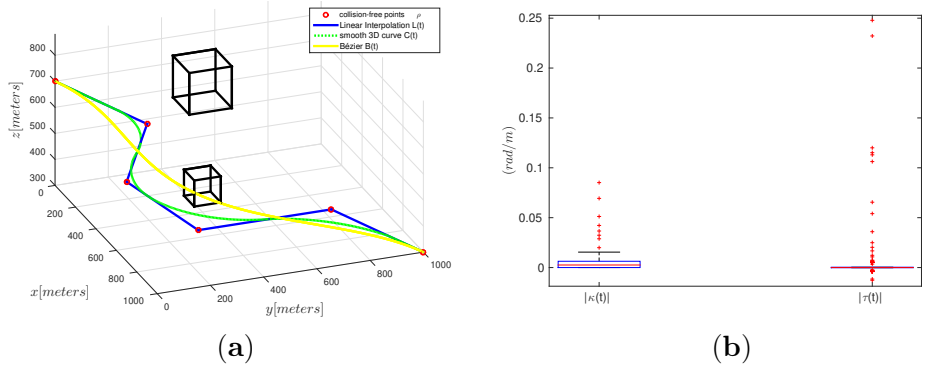



**Figura 9.8:** Example of a 3D environment cluttered of obstacles. (**a**) Reconstruction of 3D trajectories, where the obstacles are the black boxes; the red circles show the collision-free points; and the green line shows the final 3D trajectory $\boldsymbol{C}(t)$. The yellow line represents the Bézier $B(t)$ curve. (**b**) Geometric averages of the variables $\kappa$ and $\tau$ of the final path.

To describe the different groups of trajectories generated by $\boldsymbol{L}(t)$, $\boldsymbol{C}(t)$ or $B(t)$ displayed in Figure 9.9, Table 9.2 shows the flight results from the init to goal point in terms of distances; according to one of the results of each environment set by Table 9.1. It can be seen that the set of greater distances corresponding to the path in the form of the straight line marked by $\boldsymbol{L}(t)$, $\boldsymbol{C}(t)$ reduces the distance by $\boldsymbol{L}(t)$. As $B(t)$ makes an approximation (as a mathematical expression) between the set of $\rho$ of each environment, its route is the shortest. The column *"EAA Error (meters)"* shows the approximate absolute error $EAA = \frac{1}{n}\sum_{i=1}^{n}|A - B|$, where $A = \boldsymbol{L}(t)$ and $B = \boldsymbol{C}(t) \wedge B = B(t)$. Therefore, the results of the column *EAA Error (meters)* show a greater approximation by $\boldsymbol{C}(t)$ and which results in a better avoidance of obstacles.

**Tabla 9.2:** Flight distance of the curves. The column *"Flight distance (meters)"* shows the distance in meters at the initial and final free collision points marked by $\rho$. The column *"EAA Error (meters)"* shows the average error in meters along the trajectories.

| **Env.** | **Flight Distance [meters]** | | | **EAA Error [meters]** | |
|---|---|---|---|---|---|
| | ***L(t)*** | ***C(t)*** | ***B(t)*** | ***L(t) vs C(t)*** | ***L(t) vs B(t)*** |
| ***#1*** | 182.929355 | 174.002834 | 148.911388 | 0.622684 | 3.248545 |
| ***#2*** | 1728.757868 | 1610.781941 | 1453.060601 | 17.234613 | 41.453691 |
| ***#3*** | 1863.391222 | 1721.505017 | 1526.055284 | 14.600159 | 56.678212 |
| ***#4*** | 1936.078758 | 1860.263202 | 1772.944453 | 9.871725 | 36.617234 |
| ***#5*** | 1873.814514 | 1839.965587 | 1743.723244 | 9.891240 | 36.614752 |

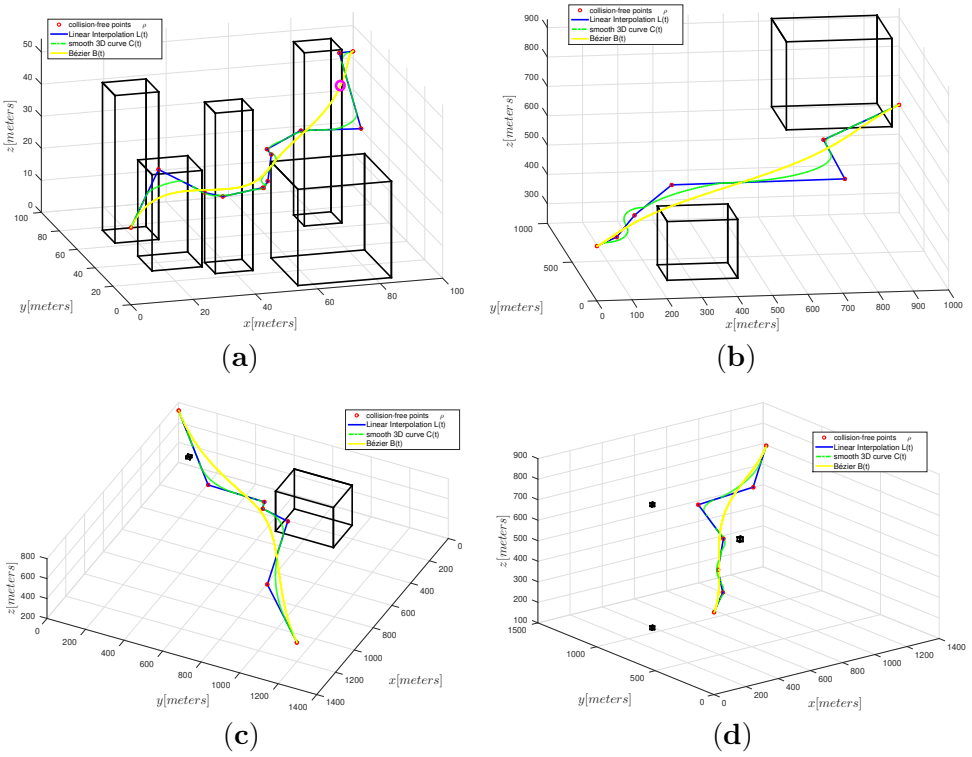


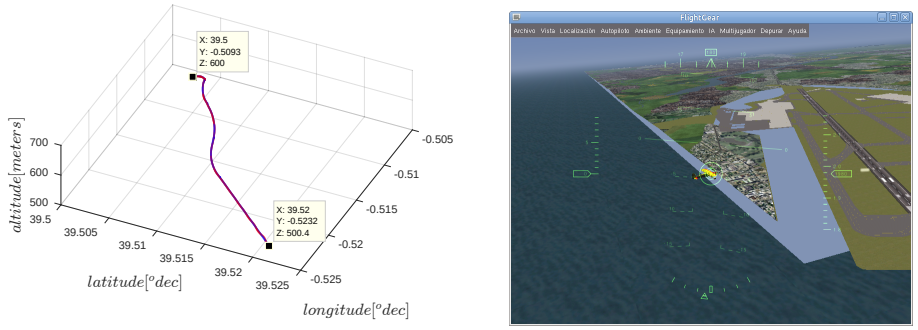

**Figura 9.9:** Additional 3D environment experiments in which the obstacles are the black boxes and the green line shows the final 3D trajectory $\boldsymbol{C}(t)$; the yellow line represents the Bézier $B(t)$ curve. (**a**) (Table 9.1—Environment #1.) It represents an unstructured environment with different buildings, where a collision between $B(t)$ and a building (collision marked as a circumference of magenta color) can be seen. (**b**) (Table 9.1—Environment #2.) 3D environment with two obstacles of different sizes. (**c**) (Table 9.1—Environment #4.) 3D environment with two obstacles of different sizes. (**d**) (Table 9.1—Environment #5.) 3D environment with three small aerial obstacles.

Similarly, Table 9.3 shows a set of results referring to the five environments analyzed. The averages of $\kappa$ and $\tau$ generated along each smooth curve shows that $B(t)$ exceeds $\boldsymbol{C}(t)$. However, in the first environment there is a collision caused by the $B(t)$ curve.

**Tabla 9.3:** Average results of $\kappa$ and $\tau$ along the curves $\boldsymbol{C}(t)$ and $B(t)$. The column "Collision" indicates collision (x) or no collision (o) of a curve against an obstacle.

| Env. | *Curve* | $\kappa$ | $\tau$ | *Collision* |
|---|---|---|---|---|
| *#1* | $\boldsymbol{C(t)}$ | 0.157961 | 0.185973 | o |
| | $\boldsymbol{B(t)}$ | 0.019513 | 0.092539 | x |
| *#2* | $\boldsymbol{C(t)}$ | 0.007138 | 0.159732 | o |
| | $\boldsymbol{B(t)}$ | 0.001082 | 0.006652 | o |
| *#3* | $\boldsymbol{C(t)}$ | 0.004556 | 0.185806 | o |
| | $\boldsymbol{B(t)}$ | 0.001068 | 0.004442 | o |
| *#4* | $\boldsymbol{C(t)}$ | 0.003445 | 0.574121 | o |
| | $\boldsymbol{B(t)}$ | 0.000812 | 0.003332 | o |
| *#5* | $\boldsymbol{C(t)}$ | 0.004515 | 0.135183 | o |
| | $\boldsymbol{B(t)}$ | 0.000643 | 0.004253 | o |

Finally, the results produced by the simulation of the UAV *kaddet 2400* "matlab/simulink/flightGear" over the environment #3 are shown in Figure 9.10. The geodesic coordinates of Figure 9.10(a) are expressed in decimal degrees; in this example the flight starts with an altitude of 500.4 m, and after the maneuvers performed by the UAV, it reaches a new altitude of 603.1 m. Figure 9.10(b) shows the UAV model in flight using FlightGear Simulator as visualization platform.




(**a**) (**b**)

**Figura 9.10:** Simulation of the UAV flight simulink, where the starting point is [39.52, -0.5232, 500] and the target point is [39.50, -0.5093, 600]. (**a**) Flight of the UAV, where the blue line is the path calculated from the process described and the red line is the actual path of the UAV. (**b**) View from a model perspective.

## 9.6 Conclusions and Future Works

This paper describes an approach to the generation of a continuous 3D smooth path that enables consideration of the operational constraints of fixed-wing UAVs.

Firstly, the document describes the formulation of the problem by defining two types of segments within the trajectory: the $S$ segments as a set of sphere segments that ensure a continuous and minimum curvature profile, and then the definition of $L$ segments that generally connect $S$.

Next, we proposed the resolution of an MOP problem to obtain the numerical values of the parameters of the trajectory, given that the problem has infinite feasible solutions. When solving an MOP problem, the DM stage is essential to finally select the desired point from the Pareto set of optimal solutions.

It is important to remember that with methods such as classic Bézier or B-splines curves, you can define the number of samples along the path. However, the distance measured between one point and the next is not the same or even close (the difference can be large). These types of curves are useful in relatively simple environments (few obstacles); however, as the number of obstacles grows, the control points increase due to trajectory planning. Consequently, the construction of the curve can cause collisions. This work enables a constant approach distance between pairs of contiguous points.

The kinematic constraints of UAVs have been considered in this work, in the same way that dynamic constraints could be calculated mathematically. However, an important consideration that can enhance the generation of new trajectories in a new job is to increase the optimization criterion by improving variables such as energy consumption or incomplete data in dynamic environments.

Connected with the results shown in this paper, several future works arise. For example, integration of dynamic obstacles (UAVs swarms or other aircraft systems) into the flying area. From the optimization point of view, the proposal can be improved by taking into account dynamic constraints (i.e, inertia, wind disturbances, torque forces, etc.) into the MOP problem. Similarly, new cost indexes as flight time or/and spent energy could be added to the optimization problem. Finally, implementation of the proposed algorithm under real conditions (UAV in an outdoor environment) and its application to different uses (such as satellite trajectory generation (Mclnnes 1998)) will be investigated.

# Capítulo 10

# Conclusiones y Trabajo futuro

***Resumen:*** *El trabajo de tesis realizado tiene como objetivo principal la generación de trayectorias aeronavegables óptimas para UAVs de ala fija. A continuación, se presentan algunas conclusiones sobre los resultados obtenidos, producto de la investigación realizada. Finalmente, los trabajos futuros introducen perspectivas de las varias y posibles direcciones que pueden tomarse a partir de los resultados obtenidos.*

## Conclusiones

Las contribuciones de este trabajo de tesis han sido comentadas en cada capítulo, formulando así conclusiones parciales. A continuación, se discuten algunas conclusiones generales, así como ideas para posibles futuros trabajos.

- *Estudio del estado del arte en planificación de trayectorias y movimiento con un enfoque 3D.*

  El capítulo 2 realiza un estudio y análisis literario en planificación de trayectorias, planificación de movimiento, la configuración del espacio, el muestreo del espacio, y varios algoritmos que utilizan dicho muestreo, con el objetivo de la definición de la metodología base a seguir durante el transcurso de la tesis. De esta forma, y de acuerdo con los resultados y las primeras conclusiones, se ha completado el trabajo publicado en

el capítulo 6, el cual parte de la base teórica del muestreo continuo o discreto y finaliza con la determinación de la opción más adecuada para el desarrollo del trabajo de tesis (siendo el muestreo discreto). La base determinista del muestreo discreto, ha sido utilizada en el capítulo 7 al plantear una nueva metodología de planificación de trayectorias como un *Autómata Finito Determinista Discreto*, metodología que consigue disminuir el esfuerzo computacional, así como la generación de nudos en el entorno de trabajo.

- *Definición de curvas suaves, y su transición hacia trayectorias suaves.*

  El capítulo 3 describe los conceptos de curvas suaves y continuas, con el objetivo de realizar una transición, partiendo de ellas, hasta llegar a la construcción de trayectorias tridimensionales suaves. Con esta base teórica, unida al modelo matemático del UAV *Kadett 2400* descrito en el capítulo 4, se ha contrastado los resultados en vuelo mostrados en los capítulos 7 y 8, donde se determina que las trayectorias generadas son adecuadas para la navegación del UAV de ala fija.

  El estudio del estado del arte descrito en el capítulo 3, señala que una metodología de concatenación de segmentos de línea recta y arcos circulares, con el objetivo de la generación de curvas es una opción viable para llegar a la construcción de trayectorias suaves y continuas, por lo que la segunda fase del trabajo de tesis explota esta base teórica.

- *Optimización en planificación de trayectorias tridimensional a partir de una malla discreta adaptativa.*

  El capítulo 7, realiza una división iterativa de entorno 3D de forma sintetizada y no necesita invocar ningún planificador adicional para la determinación de trayectorias 3D óptimas. Esto se alcanza al ubicar el UAV en un espacio libre y asignando posibles recompensas a los espacios libres de colisión adyacentes al UAV. Demostrando así que, asignar diversas recompensas a los espacios libres de colisión es una opción aceptable para la generación de trayectorias.

  La división del espacio tridimensional en una forma geométrica definida permite disminuir el número de puntos de control en una trayectoria generada, lo que a su vez disminuye el costo y la complejidad de los cálculos, proporcionando una solución que genera respuestas en tiempos relativamente más cortos en comparación con otras técnicas dedicadas a la generación de trayectorias similares.

- *Optimización Multiobjetivo y generación de trayectorias de vuelo factible con minimización del esfuerzo de giro.*

  A partir de los resultados obtenidos en el capítulo 7 (en forma de conjuntos de puntos de control), en el capítulo 9 se describe un enfoque para la generación de trayectorias suaves y continuas en un entorno tridimensional, que además considera las limitaciones operacionales de los vehículos aéreos no tripulados de ala fija. Esta problemática ha sido resuelta al definir un MOP que parte de base geométrica de generación de trayectorias que concatena segmentos (como se menciona en el capítulo 3) en forma de línea recta y segmentos curvos que parten de la forma de esferas, logrando así mantener valores de curvaturas y torsiones aceptables, dentro de los rangos establecidos.

  Finalmente, a partir de los resultados mostrados en el capítulo 9, en el que se consideran las restricciones cinemáticas de los UAVs, una consideración importante que podría mejorar la generación de nuevas trayectorias sería aumentar el criterio de optimización, mejorando variables como el consumo de energía o los datos incompletos en entornos dinámicos.

## Trabajo futuro

Los avances presentados en esta tesis pueden ampliarse mediante futuras investigaciones en varias áreas. En esta tesis, se asume que el UAV viaja a través del espacio aéreo y que busca mantener una velocidad constante definida. En este sentido, se podrá realizar un enfoque sobre las posibles colisiones entre UAVs en vuelo y la generación de rutas suaves simultáneas para un equipo o varios equipos de UAVs de ala fija.

Del mismo modo, la información de vuelo junto con la información del espacio aéreo, perturbaciones climatológicas y capacidades del UAV pueden ser integradas para la generación de nuevas alternativas de vuelo como un nuevo problema multiobjetivo. Integrando así un sistema dinámico de tareas que permita adaptarse a escenarios dinámicos o desconocidos.

Los futuros desarrollos de algoritmia de asignación de tareas se pueden centrar en crear estrategias de muestreo adaptativo sobre grupos de múltiples UAVs de ala fija con el objetivo de cubrir áreas más amplias sobre entornos terrestres o marinos. Del mismo modo, para un mejor desempeño una algoritmia de arquitectura distribuida se podría generar y eliminaría el requisito de un controlador centralizado, permitiendo la distribución del esfuerzo computacional.

Finalmente, la algoritmia propuesta para la navegación se puede extender como trayectorias con parámetros adicionales de tiempo, conectando puntos de trayectorias y su incidencia en lapsos de tiempo ideales que mejoren las maniobras del UAV.

No obstante, dada la aplicabilidad de la generación de trayectorias para UAVs, existen multiples vías de trabajo futuro, todas ellas interesantes y relevantes.

# Bibliografía

*20 great UAV applications areas for Drones* (2014). http://air-vid.com/wp/20-great-uav-applications-areas-drones/ (vid. pág. 94).

Abbadi, Ahmad y col. (mayo de 2012a). "Rapidly-exploring random trees: 3D planning". En: *In 18th International Conference on Soft Computing MENDEL*. IEEE, págs. 594-599. ISBN: 978-1-4673-5643-5 (vid. págs. 95, 97).

— (2012b). "Rapidly-exploring random trees: 3D planning". En: *18th International Conference on Soft Computing MENDEL*, págs. 594-599 (vid. pág. 149).

Abbena, Elsa, Simon Salamon y Alfred Gray (2017). *Modern differential geometry of curves and surfaces with Mathematica*. Chapman y Hall/CRC. ISBN: 9781584884484 (vid. págs. 155, 159, 169).

Aghababa, Mohammad Pourmahmood (oct. de 2012). "3D path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles". En: *Applied Ocean Research* 38, págs. 48-62. ISSN: 01411187. DOI: 10.1016/j.apor.2012.06.002 (vid. págs. 96, 97).

Agrawal, Manindra, Neeraj Kayal y Nitin Saxena (sep. de 2004). "PRIMES is in P". En: *Annals of Mathematics* 160.2, págs. 781-793. ISSN: 0003-486X. DOI: 10.4007/annals.2004.160.781 (vid. pág. 79).

Aguilar, Wilbert y Stephanie Morales (oct. de 2016). "3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms". En: *Electronics* 5.4, pág. 70. ISSN: 2079-9292. DOI: 10.3390/electronics5040070 (vid. págs. 95, 97, 149).

Aguilar, Wilbert G. y col. (feb. de 2017). "RRT* GL Based Optimal Path Planning for Real-Time Navigation of UAVs". En: *Soft Computing*. Vol. 17. 2, págs. 585-595. ISBN: 978-981-10-2524-2. DOI: 10.1007/978-3-319-59147-6_50 (vid. págs. 95, 97, 149).

Alhanjouri, Mohammed A y Belal Alfarra (2011). "Ant colony versus genetic algorithm based on travelling salesman problem". En: *International Journal of Computer Technology and Applications* 2.3, págs. 570-578. DOI: 20.500.12358/24509 (vid. pág. 150).

Almurib, H. A. F., P. T. Nathan y T. N. Kumar (sep. de 2011). "Control and path planning of quadrotor aerial vehicles for search and rescue". En: *SICE Annual Conference 2011*, págs. 700-705 (vid. pág. 13).

Amato, Nancy. M. y Yang Wu (1996). "A Randomized Roadmap Method". En: *IEEE International Conference on Robotics and Automation*, págs. 113-120 (vid. pág. 81).

Amorim, Rafhael y col. (2017). "Radio Channel Modeling for UAV Communication over Cellular Networks". En: *IEEE Wireless Communications Letters* 6.4, págs. 514-517. ISSN: 21622345. DOI: 10.1109/LWC.2017.2710045 (vid. pág. 2).

Angelov, Plamen (2012). *Sense and avoid in UAS: research and applications*. John Wiley & Sons (vid. pág. 63).

Archdeacon, Dan (1996). "Topological Graph Theory". En: *A survey. Congressus Numerantium* 115.18, págs. 5-54 (vid. págs. 20, 80).

Arkin, Ronald Craig, Patrick Ulam y Alan R Wagner (2011). "Moral decision making in autonomous systems: Enforcement, moral emotions, dignity, trust, and deception". En: *Proceedings of the IEEE* 100.3, págs. 571-589 (vid. pág. 12).

Bakdi, Azzeddine y col. (2017). "Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control". En: *Robotics and Autonomous Systems* 89, págs. 95-109. DOI: 10.1016/j.robot.2016.12.008 (vid. pág. 150).

Ball, Keith (1990). "Isometric embedding in lp spaces". En: *European Journal of Combinatorics* 11.4, págs. 305-311 (vid. pág. 21).

Barrientos, Antonio y col. (sep. de 2011). "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots". En: *Journal of Field Robotics* 28.5, págs. 667-689. ISSN: 15564959. DOI: 10.1002/rob.20403 (vid. pág. 13).

Barsky, B.A. y T.D. DeRose (nov. de 1989). "Geometric continuity of parametric curves: three equivalent characterizations". En: *IEEE Computer Graphics and Applications* 9.6, págs. 60-69. DOI: 10.1109/38.41470 (vid. pág. 150).

Barsky, Brian A y Tony D DeRose (1990). "Geometric continuity of parametric curves: constructions of geometrically continuous splines". En: *IEEE Computer Graphics and Applications* 10.1, págs. 60-68 (vid. pág. 32).

Beard, Randal W y Timothy W McLain (2012). *Small unmanned aircraft: Theory and practice*. Princeton university press (vid. pág. 150).

Beck, József y col. (2012). *Random and quasi-random point sets*. Vol. 138. Springer Science & Business Media (vid. pág. 23).

Berger, Marsha J y Joseph Oliger (mar. de 1984). "Adaptive mesh refinement for hyperbolic partial differential equations". En: *Journal of Computational Physics* 53.3, págs. 484-512. ISSN: 00219991. DOI: 10.1016/0021-9991(84)90073-1 (vid. pág. 98).

Besselmann, Thomas (2010). *Constrained optimal control: piecewise affine and linear parameter-varying systems*. ETH Zurich (vid. pág. 15).

Bhattacharya, Priyadarshi y Marina L Gavrilova (2007). "Voronoi diagram in optimal path planning". En: *4th International Symposium on Voronoi*

*Diagrams in Science and Engineering (ISVD 2007)*. IEEE, págs. 38-47 (vid. pág. 27).

Bhattacharya, Priyadarshi y Marina L Gavrilova (2008). "Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path". En: *IEEE Robotics & Automation Magazine* 15.2, págs. 58-66 (vid. pág. 24).

Blasco, X. y col. (oct. de 2008). "A new graphical visualization of n-dimensional Pareto front for decision-making in multiobjective optimization". En: *Information Sciences* 178.20, págs. 3908-3924. DOI: 10.1016/j.ins.2008.06.010 (vid. pág. 169).

Blyenburgh, Peter van (2006). "UAV systems: global review". En: *Conference, Amsterdam, The Netherlands* (vid. pág. 67).

Boissonnat, Jean-Daniel, André Cérézo y Juliette Leblond (1991). "Shortest paths of bounded curvature in the plane". En: *Workshop on Geometric Reasoning for Perception and Action*. Springer, págs. 3-18 (vid. pág. 41).

Borgefors, Gunilla (jun. de 1986). "Distance transformations in digital images". En: *Computer Vision, Graphics, and Image Processing* 34.3, págs. 344-371. ISSN: 0734189X. DOI: 10.1016/S0734-189X(86)80047-0 (vid. págs. 83, 124, 129).

Borgefors, Gunilla, Thomas Hartmann y Steven L Tanimoto (sep. de 1990). "Parallel distance transforms on pyramid machines: Theory and implementation". En: *Signal Processing* 21.1, págs. 61-86. ISSN: 01651684. DOI: 10.1016/0165-1684(90)90027-V (vid. pág. 83).

Boyd, Stephen, Stephen P Boyd y Lieven Vandenberghe (2004). *Convex optimization*. Cambridge university press (vid. pág. 53).

Brezak, Mišel e Ivan Petrović (2013). "Real-time approximation of clothoids with bounded error for path planning applications". En: *IEEE Transactions on Robotics* 30.2, págs. 507-515. DOI: 10.1109/TRO.2013.2283928 (vid. pág. 150).

Buniyamin, Norlida y col. (2011). "A simple local path planning algorithm for autonomous mobile robots". En: *International journal of systems applications, Engineering & development* 5.2, págs. 151-159 (vid. pág. 27).

Cai, Wenyu, Meiyan Zhang y Yahong Rosa Zheng (2017). "Task assignment and path planning for multiple autonomous underwater vehicles using 3D dubins curves". En: *Sensors* 17.7, pág. 1607 (vid. págs. 31, 38).

Calinon, Sylvain (2018). "Robot Learning with Task-Parameterized Generative Models". En: *Proc. Intl Symp. on Robotics Research*, págs. 111-126. DOI: 10.1007/978-3-319-60916-4_7 (vid. pág. 11).

Campion, Guy, Georges Bastin y Brigitte Dandrea-Novel (1996). "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots". En: *IEEE transactions on robotics and automation* 12.1, págs. 47-62 (vid. pág. 46).

Campoy, Pascual y col. (2006). "Control en el espacio de estado". En: (vid. pág. 14).

Cao, Hongxin y col. (2016). "Visualized trajectory planning of flexible redundant robotic arm using a novel hybrid algorithm". En: *Optik* 127.20, págs. 9974-9983. DOI: 10.1016/j.ijleo.2016.07.078 (vid. pág. 150).

Caselli, Stefano y Monica Reggiani (2000). "ERPP: An experience-based randomized path planner". En: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE, págs. 1002-1008 (vid. pág. 23).

Caselli, Stefano, Monica Reggiani y Roberto Rocchi (2001). "Heuristic methods for randomized path planning in potential fields". En: *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No. 01EX515)*. IEEE, págs. 426-431 (vid. pág. 23).

Cekmez, Ugur, Mustafa Ozsiginan y Ozgur Koray Sahingoz (2014). "A UAV path planning with parallel ACO algorithm on CUDA platform". En: *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, págs. 347-354 (vid. pág. 24).

Cekmez, Ugur, Mustafa Ozsiginan y Ozgur Koray Sahingoz (jun. de 2016). "Multi colony ant optimization for UAV path planning with obstacle avoidance". En: *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, págs. 47-52. ISBN: 978-1-4673-9334-8. DOI: 10.1109/ICUAS.2016.7502621 (vid. pág. 79).

Cha, Sung-Hyuk (2007). "Comprehensive survey on distance/similarity measures between probability density functions". En: *City* 1.2, pág. 1 (vid. pág. 20).

Châari, Imen y col. (2012). "SmartPATH: A hybrid ACO-GA algorithm for robot path planning". En: *2012 IEEE congress on evolutionary computation*. IEEE, págs. 1-8 (vid. pág. 24).

Chang, Hao y col. (2018). "Design of Tracked Model Vehicle Measurement and Control System Based on VeriStand and Simulink". En: *MATEC Web of Conferences*. Vol. 175. EDP Sciences, pág. 03047 (vid. pág. 72).

Chang, Seong-Ryong y Uk-Youl Huh (dic. de 2014). "A Collision-Free $G^2$ Continuous Path-Smoothing Algorithm Using Quadratic Polynomial Interpolation". En: *International Journal of Advanced Robotic Systems* 11.12, pág. 194. DOI: 10.5772/59463 (vid. pág. 151).

Chee, KY y ZW Zhong (2013). "Control, navigation and collision avoidance for an unmanned aerial vehicle". En: *Sensors and Actuators A: Physical* 190, págs. 66-76. DOI: 10.1016/j.sna.2012.11.017 (vid. pág. 149).

Chen, Xia y Jing Zhang (ago. de 2013). "The Three-Dimension Path Planning of UAV Based on Improved Artificial Potential Field in Dynamic Environment". En: *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*. Vol. 2. IEEE, págs. 144-147. ISBN: 978-0-7695-5011-4. DOI: 10.1109/IHMSC.2013.181 (vid. págs. 95, 97, 150).

Chen, Xiong y col. (2013). "A fast two-stage ACO algorithm for robotic path planning". En: *Neural Computing and Applications* 22.2, págs. 313-319 (vid. pág. 24).

Cheng, Xianyi y Changyu Yang (2008). "Robot path planning based on adaptive isolation niche genetic algorithm". En: *2008 Second International Sym-*

*posium on Intelligent Information Technology Application*. Vol. 2. IEEE, págs. 151-154. DOI: 10.1109/IITA.2008.130 (vid. pág. 150).

Cheol Chang, Myung Jin Chung y Bum Hee Lee (mar. de 1994). "Collision avoidance of two general robot manipulators by minimum delay time". En: *IEEE Transactions on Systems, Man, and Cybernetics* 24.3, págs. 517-522. ISSN: 00189472. DOI: 10.1109/21.279000 (vid. pág. 78).

Choset, Howie M y col. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press (vid. pág. 12).

Claesson, A. y col. (mayo de 2017). "Drones may be used to save lives in out of hospital cardiac arrest due to drowning". En: *Resuscitation* 114, págs. 152-156. ISSN: 03009572. DOI: 10.1016/j.resuscitation.2017.01.003. eprint: arXiv:0912.5467v3 (vid. pág. 94).

Clapham, Christopher (1992). *Diccionario Oxford de matemáticas*. Celeste (vid. pág. 36).

Clapper, J y col. (2007). "Unmanned systems roadmap 2007-2032". En: *Office of the Secretary of Defense* 188 (vid. pág. 62).

Clímaco, J y J Craveirinha (2005). *Multiple criteria decision analysis–state of the art surveys* (vid. pág. 50).

Clothier, Reece A y col. (2011). "Definition of an airworthiness certification framework for civil unmanned aircraft systems". En: *Safety science* 49.6, págs. 871-885 (vid. pág. 62).

Clough, Bruce (2002). "Metrics, Schmetrics! How Do You Track a UAV's Autonomy?" En: *1st UAV Conference*, pág. 3499 (vid. págs. 68, 69).

Coello, Carlos A Coello (2011). "Evolutionary multi-objective optimization: basic concepts and some applications in pattern recognition". En: *Mexican Conference on Pattern Recognition*. Springer, págs. 22-33 (vid. pág. 51).

Coello, Carlos A Coello y Nareli Cruz Cortés (2005). "Solving multiobjective optimization problems using an artificial immune system". En: *Genetic programming and evolvable machines* 6.2, págs. 163-190 (vid. pág. 51).

Coello, Carlos A Coello y Gary B Lamont (2004). *Applications of multi-objective evolutionary algorithms*. Vol. 1. World Scientific (vid. pág. 51).

Coello, Carlos A Coello, Gary B Lamont, David A Van Veldhuizen y col. (2007). *Evolutionary algorithms for solving multi-objective problems*. Vol. 5. Springer (vid. pág. 50).

Coello Coello, C. A., G. Toscano Pulido y E. Mezura Montes (2005). "Current and Future Research Trends in Evolutionary Multiobjective Optimization". En: *Information Processing with Evolutionary Algorithms*. Springer-Verlag, págs. 213-231. DOI: 10.1007/1-84628-117-2_15 (vid. pág. 165).

Cordero, Luis A, Marisa Fernandez y Alfred Gray (1995). *Geometria differencial de curvas y superfices*. Addison-Wesley IberoAmericana (vid. pág. 35).

Courbon, Jonathan y col. (2010). "Vision-based navigation of unmanned aerial vehicles". En: *Control Engineering Practice* 18.7, págs. 789-799. DOI: 10.1016/j.conengprac.2010.03.004 (vid. pág. 149).

Cuerno-Rejado, Cristina y col. (2016). "Evolución histórica de los vehículos aéreos no tripulados hasta la actualidad". En: *Dyna (Spain)* 91.3, pág. 332. ISSN: 19891490. DOI: 10.6036/7781 (vid. pág. 12).

Dalamagkidis, Konstantinos (2015). "Classification of uavs". En: *Handbook of unmanned aerial vehicles*, págs. 83-91 (vid. pág. 68).

De Boor, Carl y col. (1978). *A practical guide to splines*. Vol. 27. springer-verlag New York (vid. págs. 43, 44).

De Filippis, Luca, Giorgio Guglieri y Fulvia Quagliotti (ene. de 2012). "Path Planning Strategies for UAVS in 3D Environments". En: *Journal of Intelligent & Robotic Systems* 65.1-4, págs. 247-264. ISSN: 0921-0296. DOI: 10.1007/s10846-011-9568-2 (vid. pág. 96).

De Lorenzis, Laura, Peter Wriggers y Thomas JR Hughes (2014). "Isogeometric contact: a review". En: *GAMM-Mitteilungen* 37.1, págs. 85-123. DOI: 10.1002/gamm.201410005 (vid. pág. 151).

Delingette, Hervé, Martial Hebert y Katsushi Ikeuchi (1991). "Trajectory generation with curvature constraint based on energy minimization". En: (vid. págs. 39, 41).

Denny, Jory y Nancy M Amatoo (2013a). "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension". En: *Algorithmic Foundations of Robotics X*. Springer, págs. 297-312 (vid. pág. 24).

— (2013b). "Toggle PRM: A Coordinated Mapping of C-Free and C-Obstacle in Arbitrary Dimension". En: vol. 86, págs. 297-312. ISBN: 978-3-642-36278-1. DOI: 10.1007/978-3-642-36279-8_18 (vid. págs. 95, 97, 150).

Díaz, Iñaki, Jorge Juan Gil y Emilio Sánchez (2011). "Lower-Limb Robotic Rehabilitation: Literature Review and Challenges". En: *Journal of Robotics* 2011.i, págs. 1-11. ISSN: 1687-9600. DOI: 10.1155/2011/759764 (vid. pág. 11).

Diebel, James (nov. de 2006). "Representing attitude: Euler angles, unit quaternions, and rotation vectors". En: *Matrix* 58.15-16, págs. 1-35 (vid. pág. 126).

Dijkstra, E W (dic. de 1959). "A note on two problems in connexion with graphs". En: *Numerische Mathematik* 1.1, págs. 269-271. ISSN: 0029-599X. DOI: 10.1007/BF01386390 (vid. págs. 87, 96, 103, 104).

*Disposición 15721 del BOE núm. 316 de 2017 - BOE.es* (s.f.). https://www.boe.es/boe/dias/2017/12/29/pdfs/BOE-A-2017-15721.pdf (accessed on 28 December 2018) (vid. pág. 116).

Dixon, Cory y Eric W Frew (2007). "Decentralized extremum-seeking control of nonholonomic vehicles to form a communication chain". En: *Advances in cooperative control and optimization*. Springer, págs. 311-322. DOI: 10.1007/978-3-540-74356-9_19 (vid. pág. 148).

Duan, Haibin y Peixin Qiao (2014). "Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning". En: *International Journal of Intelligent Computing and Cybernetics* 7.1, págs. 24-37. DOI: 10.1108/ijicc.2009.39802aaa.001 (vid. pág. 150).

Duan, Haibin, Yaxiang Yu y col. (2010). "Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm". En: *Simulation Modelling Practice and Theory* 18.8, págs. 1104-1115 (vid. págs. 24, 96, 97).

Dubé, R. y col. (oct. de 2016). "3D localization, mapping and path planning for search and rescue operations". En: *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, págs. 272-273. DOI: 10.1109/SSRR.2016.7784311 (vid. pág. 13).

Dubins, Lester E (1957). "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents". En: *American Journal of mathematics* 79.3, págs. 497-516 (vid. págs. 39, 150).

Efstratiadis, Andreas y Demetris Koutsoyiannis (2010). "One decade of multi-objective calibration approaches in hydrological modelling: a review". En: *Hydrological Sciences Journal–Journal Des Sciences Hydrologiques* 55.1, págs. 58-78 (vid. pág. 51).

Ehrgott, Matthias (2012). "Vilfredo Pareto and multi-objective optimization". En: *Doc. math*, págs. 447-453 (vid. pág. 54).

Elbanhawi, Mohamed y Milan Simic (2014). "Sampling-Based Robot Motion Planning: A Review". En: *IEEE Access* 2, págs. 56-77. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2302442 (vid. pág. 95).

Endo, Nobutsuna y col. (2008). "Development of whole-body emotion expression humanoid robot". En: *2008 IEEE International Conference on Robotics and Automation*. IEEE, págs. 2140-2145 (vid. pág. 12).

Evestedt, Niclas y col. (2015). "Sampling recovery for closed loop rapidly expanding random tree using brake profile regeneration". En: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, págs. 101-106 (vid. pág. 24).

Fadaee, M y MAM Radzi (2012). "Multi-objective optimization of a stand-alone hybrid renewable energy system by using evolutionary algorithms: A review". En: *Renewable and sustainable energy reviews* 16.5, págs. 3364-3369 (vid. pág. 51).

Fagnant, Daniel J y Kara Kockelman (2015). "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations". En: *Transportation Research Part A: Policy and Practice* 77, págs. 167-181. DOI: 10.1016/j.tra.2015.04.003 (vid. pág. 148).

Fang, Zheng, Chengzhi Luan y Zhiming Sun (2017). "A 2D Voronoi-Based Random Tree for Path Planning in Complicated 3D Environments". En: vol. 531, págs. 433-445. ISBN: 978-3-319-48035-0. DOI: 10.1007/978-3-319-48036-7_31 (vid. págs. 95, 97, 150).

Farin, Gerald (2014). *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier (vid. pág. 152).

Farin, Gerald E y Gerald Farin (2002). *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann (vid. pág. 40).

Ferguson, Dave y Anthony Stentz (s.f.). "Field D*: An Interpolation-Based Path Planner and Replanner". En: *Robotics Research*. Berlin, Heidelberg: Springer Berlin Heidelberg, págs. 239-253. ISBN: 9783540481102. DOI: 10.1007/978-3-540-48113-3_22 (vid. pág. 96).

Fishpool, M (2010). *International military and civilian Unmanned Aerial Vehicle survey*. Inf. téc. Tech. rep., Socolofi Research (vid. pág. 66).

Fleury, Sara y col. (1995). "Primitives for smoothing mobile robot trajectories". En: *IEEE transactions on robotics and automation* 11.3, págs. 441-448. DOI: 10.1109/70.388788 (vid. pág. 150).

Fonseca, Carlos M y Peter J Fleming (1998). "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. II. Application example". En: *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and humans* 28.1, págs. 38-47 (vid. pág. 57).

Fraichard, Thierry y Alexis Scheuer (2004). "From Reeds and Shepp's to continuous-curvature paths". En: *IEEE Transactions on Robotics* 20.6, págs. 1025-1035. DOI: 10.1109/TRO.2004.833789 (vid. pág. 151).

Gafurov, Salimzhan A y Evgeniy V Klochkov (2015). "Autonomous unmanned underwater vehicles development tendencies". En: *Procedia Engineering* 106, págs. 141-148. DOI: 10.1016/j.proeng.2015.06.017 (vid. pág. 149).

García-Martínez, Carlos, Oscar Cordón y Francisco Herrera (2007). "A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP". En: *European Journal of Operational Research* 180.1, págs. 116-148 (vid. pág. 51).

Garrido, Santiago y col. (2006). "Path planning for mobile robot navigation using voronoi diagram and fast marching". En: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, págs. 2376-2381 (vid. pág. 27).

Gautam, S. Aditya y Nilmani Verma (sep. de 2014). "Path planning for unmanned aerial vehicle based on genetic algorithm & artificial neural network in 3D". En: *2014 International Conference on Data Mining and Intelligent Computing (ICDMIC)*. IEEE, págs. 1-5. ISBN: 978-1-4799-4674-7. DOI: 10.1109/ICDMIC.2014.6954257 (vid. págs. 96, 97).

Gim, Suhyeon y col. (2017). "Clothoids composition method for smooth path generation of car-like vehicle navigation". En: *Journal of Intelligent & Robotic Systems* 88.1, págs. 129-146 (vid. pág. 43).

Girbés, Vicent, Gloria Vanegas y Leopoldo Armesto (2019). "Clothoid-Based Three-Dimensional Curve for Attitude Planning". En: *Journal of Guidance, Control, and Dynamics*, págs. 1-13. DOI: 10.2514/1.G003551 (vid. pág. 151).

Girbés Juan, Vicent (2016). "Clothoid-based Planning and Control in Intelligent Vehicles (Autonomous and Manual-Assisted Driving)". Tesis doct. (vid. pág. 42).

Goel, Utkarsh y col. (2018). "Three Dimensional Path Planning for UAVs in Dynamic Environment using Glow-worm Swarm Optimization". En: *Procedia Computer Science* 133, págs. 230-239. ISSN: 18770509. DOI: 10.1016/j.procs.2018.07.028 (vid. págs. 96, 97).

Gómez, Carlos Munuera y Fernando Torres (2006). "Sobre curvas algebraicas y códigos correctores". En: *Gaceta de la Real Sociedad Matematica Española* 9.1, págs. 203-222 (vid. pág. 35).

Graham, Andrew (2002). "HyperPhysics". En: *The Physics Teacher* 40.5, págs. 318-318 (vid. pág. 43).

Guidance, Interim Operational Approval (2008). "08-01, Unmanned Aircraft Systems Operations in the US National Airspace System". En: *Federal Aviation Administration* (vid. pág. 62).

Guoqing Zhou y Deyan Zang (2007). "Civil UAV system for earth observation". En: *2007 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, págs. 5319-5322. ISBN: 978-1-4244-1211-2. DOI: 10.1109/IGARSS.2007.4424063 (vid. pág. 78).

Hahn, Peter (1978). "Haar measure for measure groupoids". En: *Transactions of the American Mathematical Society* 242, págs. 1-33 (vid. pág. 23).

Hales, Thomas C (2007a). "Jordans proof of the Jordan curve theorem". En: *Studies in logic, grammar and rhetoric* 10.23, págs. 45-60 (vid. pág. 36).

— (2007b). "The Jordan curve theorem, formally and informally". En: *The American Mathematical Monthly* 114.10, págs. 882-894 (vid. pág. 36).

Hameed, I. A. (2014). "Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain". En: *Journal of Intelligent and Robotic Systems: Theory and Applications* 74.3-4, págs. 965-983. ISSN: 15730409. DOI: 10.1007/s10846-013-9834-6 (vid. pág. 13).

Hart, Peter E y J Nils (1968). "A formal basis for the Heuristic Determination of minimum cost paths". En: *IEEE transactions on Systems Science and Cybernetics* 4.2, págs. 100-107 (vid. págs. 87, 96, 104).

Hartman, Bruce I, Yutaka Kanayama y Terence Smith (1989). "Model and Sensor Based Precise Navigation by an Autonomous Mobile Robot". En: *Advanced Robotics: 1989*. Springer, págs. 98-109 (vid. pág. 39).

Hasbestan, Jaber J. e Inanc Senocak (2018). "Binarized-octree generation for Cartesian adaptive mesh refinement around immersed geometries". En: *Journal of Computational Physics* 368, págs. 179-195. ISSN: 10902716. DOI: 10.1016/j.jcp.2018.04.039 (vid. pág. 98).

He, Yufeng y col. (mayo de 2013). "Path planning for indoor UAV based on Ant Colony Optimization". En: *2013 25th Chinese Control and Decision Conference (CCDC)*. IEEE, págs. 2919-2923. ISBN: 978-1-4673-5534-6. DOI: 10.1109/CCDC.2013.6561444 (vid. págs. 96, 97).

Hempe, D (2006). "Unmanned aircraft systems in the united states". En: *US/Europe international safety conference* (vid. pág. 69).

Heping Chen, T. Fuhlbrigge y Xiongzi Li (ago. de 2008). "Automated industrial robot path planning for spray painting process: A review". En: *2008 IEEE International Conference on Automation Science and Engineering*, págs. 522-527. DOI: 10.1109/COASE.2008.4626515 (vid. pág. 13).

Herlik, E (2010). *Unmanned Aerial Vehicles (UAVs) for commercial applications global market & technologies outlook 2011–2016*. Inf. téc. Technical report, Market Intel Group LLC (vid. pág. 66).

Hernandez, Kevin, Bladimir Bacca y Breyner Posso (feb. de 2017). "Multigoal Path Planning Autonomous System for Picking up and Delivery Tasks in Mobile Robotics". En: *IEEE Latin America Transactions* 15.2, págs. 232-238. ISSN: 1548-0992. DOI: 10.1109/TLA.2017.7854617 (vid. pág. 95).

Hernández, Eugenio, Ma Angeles Zurro y María Jesús Vazquez (2012). *Álgebra lineal y Geometría*. Pearson (vid. pág. 95).

Herrero, J M y col. (2007). "Well-Distributed Pareto Front by Using the $\epsilon^{\nearrow}-MOGA$ Evolutionary Algorithm". En: *Computational and Ambient Intelligence*. Springer Berlin Heidelberg, págs. 292-299. DOI: 10.1007/978-3-540-73007-1_36 (vid. págs. 151-153).

Hobby, John D (1986). "Smooth, easy to compute interpolating splines". En: *Discrete & Computational Geometry* 1.2, págs. 123-140 (vid. pág. 31).

Honig, Eli, Engelbert L. Schucking y C. V. Vishveshwara (jun. de 1974). "Motion of charged particles in homogeneous electromagnetic fields". En: *Journal of Mathematical Physics* 15.6, págs. 774-781. DOI: 10.1063/1.1666728 (vid. pág. 155).

Huang, Han-Pang y Shu-Yun Chung (2004). "Dynamic visibility graph for path planning". En: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE, págs. 2813-2818 (vid. pág. 27).

Huang, Hsu-Chih y Ching-Chih Tsai (2011). "Global path planning for autonomous robot navigation using hybrid metaheuristic GA-PSO algorithm". En: *SICE Annual Conference 2011*. IEEE, págs. 1338-1343 (vid. pág. 24).

Huh, Uk-Youl y Seong-Ryong Chang (2014). "A $G^2$ continuous path-smoothing algorithm using modified quadratic polynomial interpolation". En: *International Journal of Advanced Robotic Systems* 11.2, pág. 25. DOI: 10.5772/57340 (vid. pág. 151).

Hull, David G (2007). *Fundamentals of airplane flight mechanics*. Also available as https://scholar.google.co.uk/scholar?q=fundamental+of+airplane+flight+mechanics+david+hull&btnG=&hl=en&as_sdt=0,5#0. Springer. ISBN: 9783540465713 (vid. pág. 149).

Isaacs, Jason y João Hespanha (feb. de 2013). "Dubins Traveling Salesman Problem with Neighborhoods: A Graph-Based Approach". En: *Algorithms* 6.1, págs. 84-99 (vid. pág. 151).

Ismail, AT, Alaa Sheta y Mohammed Al-Weshah (2008). "A mobile robot path planning using genetic algorithm in static environment". En: *Journal of Computer Science* 4.4, págs. 341-344 (vid. pág. 24).

Iswanto, Iswanto, Oyas Wahyunggoro y Adha Imam Cahyadi (2016). "Quadrotor Path Planning Based on Modified Fuzzy Cell Decomposition Algorithm". En: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 14.2, pág. 655. ISSN: 1442-2042. DOI: 10.12928/TELKOMNIKA.v14i1.2989 (vid. págs. 96, 97).

Iyer, B. R. y C. V. Vishveshwara (jul. de 1988). "The Frenet-Serret formalism and black holes in higher dimensions". En: *Classical and Quantum Gravity* 5.7, págs. 961-970. DOI: 10.1088/0264-9381/5/7/005 (vid. pág. 155).

Jackson, Paul (2004). "Beech King Air B200". En: *Jane's All the World's Aircraft 2003-2004, 2004. ISBN 0-7106-2537* 5 (vid. pág. 63).

Jacobs, Paul y John Canny (1993). "Planning smooth paths for mobile robots". En: *Nonholonomic Motion Planning*. Springer, págs. 271-342 (vid. pág. 42).

Jacobs, Paul, Greg Heinzinger y col. (1989). "Planning Guaranteed Near Time-Optimal Trajectories for a Manipulator in a Cluttered Workspace". En: *International Workshop on Sensorial Integration for Industrial Robots: Architectures and Applications, Zaragoza, Spain* (vid. pág. 39).

Jaillet, Léonard, Juan Cortés y Thierry Siméon (2008). "Transition-based RRT for path planning in continuous cost spaces". En: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, págs. 2145-2150 (vid. pág. 24).

Jenkins, Darryl y Bijan Vasigh (2013). "The Economic Impact of Unmanned Aircraft Systems Integration in the United States". En: *Association for Unmanned Vehicle Systems International* March, págs. 1-40 (vid. pág. 2).

Jones, Joseph L (2006). "Robots at the tipping point: the road to iRobot Roomba". En: *IEEE Robotics & Automation Magazine* 13.1, págs. 76-78 (vid. pág. 11).

Jun, Jae-Yun, Jean-Philippe Saut y Faiz Benamar (2016). "Pose estimation-based path planning for a tracked mobile robot traversing uneven terrains". En: *Robotics and Autonomous Systems* 75, págs. 325-339. DOI: 10.1016/j.robot.2015.09.014 (vid. pág. 149).

Justh, Eric W and Krishnaprasad, Perinkulam S (oct. de 2002). *A Simple Control Law for UAV Formation Flying*. Inf. téc. 10, págs. 1-35 (vid. pág. 124).

Kaluđer, Hrvoje, Mišel Brezak e Ivan Petrović (2011). "A visibility graph based method for path planning in dynamic environments". En: *2011 Proceedings*

*of the 34th International Convention MIPRO*. IEEE, págs. 717-721 (vid. pág. 27).

Kanayama, Yutaka (1986). "Trajectory generation for mobile robots". En: *Robotics Research*, págs. 333-340 (vid. pág. 41).

Kanayama, Yutaka J y Bruce I Hartman (1997). "Smooth local-path planning for autonomous vehicles1". En: *The International Journal of Robotics Research* 16.3, págs. 263-284 (vid. pág. 41).

Karlovitz, LA (1970). "Construction of nearest points in the Lp, p even, and L norms. I". En: *Journal of Approximation Theory* 3.2, págs. 123-127 (vid. pág. 21).

Khuswendi, Taufik, Hilwadi Hindersah y Widyawardana Adiprawita (jul. de 2011). "UAV path planning using potential field and modified receding horizon A* 3D algorithm". En: *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*. July. IEEE, págs. 1-6. ISBN: 978-1-4577-0753-7. DOI: 10.1109/ICEEI.2011.6021579 (vid. págs. 95, 97, 150).

Kiciak, Przemysław (2016). *Geometric Continuity of Curves and Surfaces*. Vol. 8. 3, págs. 1-249. ISBN: 9781627059053. DOI: 10.2200/s00729ed1v01y201608vcp025 (vid. pág. 39).

Kim, Hanguen y col. (jul. de 2014). "Angular rate-constrained path planning algorithm for unmanned surface vehicles". En: *Ocean Engineering* 84, págs. 37-44. DOI: 10.1016/j.oceaneng.2014.03.034 (vid. pág. 151).

Klančar, Gregor e Igor Škrjanc (2010). "A case study of the collision-avoidance problem based on Bernstein-Bézier path tracking for multiple robots with known constraints". En: *Journal of Intelligent and Robotic Systems: Theory and Applications*. ISSN: 09210296. DOI: 10.1007/s10846-010-9417-8 (vid. pág. 39).

Klein, Vladislav y Eugene A Morelli (2006). *Aircraft system identification: theory and practice*. American Institute of Aeronautics y Astronautics Reston, VA (vid. pág. 74).

Kohlbrecher, Stefan y col. (nov. de 2011). "A flexible and scalable SLAM system with full 3D motion estimation". En: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. Vol. 32. 5. IEEE, págs. 155-160. ISBN: 978-1-61284-769-6. DOI: 10.1109/SSRR.2011.6106777 (vid. pág. 95).

Kolmanovsky, Ilya and McClamroch, N Harris (1995). "Developments in nonholonomic control problems". En: *IEEE Control systems magazine* 15.6, págs. 20-36 (vid. pág. 79).

Krotkov, Eric y Martial Hebert (1995). "Mapping and positioning for a prototype lunar rover". En: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Vol. 3. IEEE, págs. 2913-2919. DOI: 10.1109/ROBOT.1995.525697 (vid. pág. 148).

Kroumov, Valeri y Jianli Yu (mar. de 2009). "3D path planning for mobile robots using annealing neural network". En: *2009 International Conference on Networking, Sensing and Control*. IEEE, págs. 130-135. ISBN: 978-1-4244-3491-6. DOI: 10.1109/ICNSC.2009.4919259 (vid. págs. 96, 97).

Kunchev, Voemir y col. (2006). "Path planning and obstacle avoidance for autonomous mobile robots: A review". En: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, págs. 537-544 (vid. pág. 12).

Kwangjin Yang y Salah Sukkarieh (sep. de 2008). "3D smooth path planning for a UAV in cluttered natural environments". En: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, págs. 794-800. ISBN: 978-1-4244-2057-5. DOI: 10.1109/IROS.2008.4650637 (vid. págs. 125, 133).

Kyriakidis, Miltos, Riender Happee y Joost CF de Winter (2015). "Public opinion on automated driving: Results of an international questionnaire among 5000 respondents". En: *Transportation research part F: traffic psychology and behaviour* 32, págs. 127-140. DOI: 10.1016/j.trf.2015.04.014 (vid. pág. 148).

Lafuente, J (1998). "GEOMETRIA DIFERENCIAL DE CURVAS EN EL PLANO." En: (vid. pág. 37).

Latombe, Jean-Claude (2012). *Robot motion planning*. Vol. 124. Springer Science & Business Media (vid. pág. 12).

Laumanns, Marco y col. (sep. de 2002). "Combining Convergence and Diversity in Evolutionary Multiobjective Optimization". En: *Evolutionary Computation* 10.3, págs. 263-282. DOI: 10.1162/106365602760234108 (vid. pág. 165).

LaValle, Steven M (1998). "Rapidly-exploring random trees: A new tool for path planning". En: DOI: doi.org/10.1.1.35.1853 (vid. págs. 81, 124).

Lavalle, Steven M (2006). "PLANNING ALGORITHMS". En: http://planning.cs.uiuc.edu/ (vid. pág. 95).

LaValle, Steven M. (2006). *Planning Algorithms*. Cambridge: Cambridge University Press, págs. 1-826. ISBN: 9780511546877. DOI: 10.1017/CBO9780511546877 (vid. págs. 12, 14, 18, 22, 28).

LaValle, Steven M and Kuffner Jr, James J (2000). *Rapidly-Exploring Random Trees: Progress and Prospets*. Citeseer (vid. págs. 81, 124).

Lazard, Sylvain, John Reif y Hongyan Wang (1998). "The Complexity of the Two Dimensional Curvature-Constrained Shortest-Path Problem". En: *Proceedings of the Third International Workshop on the Algorithmic Foundations of Robotics*, págs. 49-57 (vid. pág. 156).

Li, Boyang y col. (2016). "Development and testing of a two-UAV communication relay system". En: *Sensors (Switzerland)* 16.10. ISSN: 14248220. DOI: 10.3390/s16101696 (vid. págs. 2, 149).

Li, Jin y Youngnam Han (mar. de 2017). "Optimal Resource Allocation for Packet Delay Minimization in Multi-Layer UAV Networks". En: *IEEE Communications Letters* 21.3. http://ieeexplore.ieee.org/document/7738405/, págs. 580-583. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2016.2626293 (vid. pág. 94).

Li, Qianru y col. (2014a). "Improved PRM method of low altitude penetration trajectory planning for UAVs". En: *Proceedings of 2014 IEEE Chinese Gui-*

*dance, Navigation and Control Conference*. IEEE, págs. 2651-2656 (vid. pág. 24).

Li, Qianru y col. (ago. de 2014b). "Improved PRM method of low altitude penetration trajectory planning for UAVs". En: *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*. IEEE, págs. 2651-2656. ISBN: 978-1-4799-4699-0. DOI: 10.1109/CGNCC.2014.7007587 (vid. págs. 95, 97, 150).

Liang, Yuming y Lihong Xu (2009). "Global path planning for mobile robot based genetic algorithm and modified simulated annealing algorithm". En: *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*. ACM, págs. 303-308. DOI: 10.1145/1543834.1543875 (vid. pág. 150).

Lifen, Liu y col. (2016). "Path planning for UAVS based on improved artificial potential field method through changing the repulsive potential function". En: *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*. IEEE, págs. 2011-2015 (vid. pág. 150).

Likhachev, Maxim, Dave Ferguson y col. (sep. de 2008). "Anytime search in dynamic graphs". En: *Artificial Intelligence* 172.14, págs. 1613-1643. ISSN: 00043702. DOI: 10.1016/j.artint.2007.11.009 (vid. pág. 87).

Likhachev, Maxim, Geoff Gordon y Sebastian Thrun (2014). "ARA *: Anytime A * with Provable Bounds on". En: *Science*, págs. 767-774 (vid. pág. 87).

Liu, Jianhua y col. (2017). "An improved ant colony algorithm for robot path planning". En: *Soft Computing* 21.19, págs. 5829-5839. DOI: 10.1007/s00500-016-2161-7 (vid. pág. 150).

Liu, Shifei, Yanhui Wei y Yanbin Gao (abr. de 2012). "3D path planning for AUV using fuzzy logic". En: *Computer Science and Information Processing (CSIP)*, págs. 599-603. ISSN: 0031-899X (vid. págs. 96, 97).

Liu Lifen y col. (ago. de 2016). "Path planning for UAVS based on improved artificial potential field method through changing the repulsive potential function". En: *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*. IEEE, págs. 2011-2015. ISBN: 978-1-4673-8318-9. DOI: 10.1109/CGNCC.2016.7829099 (vid. págs. 95, 97).

Lotov, Alexander V y Kaisa Miettinen (2008). "Visualizing the Pareto frontier". En: *Multiobjective optimization*. Springer, págs. 213-243 (vid. pág. 50).

Lozano-Pérez, Tomás y Michael A Wesley (1979). "An algorithm for planning collision-free paths among polyhedral obstacles". En: *Communications of the ACM* 22.10, págs. 560-570 (vid. pág. 18).

Ma, Yingchong, Gang Zheng y Wilfrid Perruquetti (2013). "Cooperative path planning for mobile robots based on visibility graph". En: *Proceedings of the 32nd Chinese Control Conference*. IEEE, págs. 4915-4920 (vid. pág. 27).

Mac, Thi Thoa y col. (dic. de 2016). "Heuristic approaches in robot path planning: A survey". En: *Robotics and Autonomous Systems* 86, págs. 13-28. ISSN: 09218890. DOI: 10.1016/j.robot.2016.08.001 (vid. pág. 96).

Majeed, Abdul y Sungchang Lee (2018). "A fast global flight path planning algorithm based on space circumscription and sparse visibility graph for unmanned aerial vehicle". En: *Electronics* 7.12, pág. 375 (vid. pág. 27).

Malekzadeh, Milad S y col. (2014). "Learning by imitation with the STIFF-FLOP surgical robot: a biomimetic approach inspired by octopus movements". En: *Robotics and Biomimetics* 1.1, pág. 13. ISSN: 2197-3768. DOI: 10.1186/preaccept-1725134953134311 (vid. pág. 11).

Mansouri, S Afshin, David Gallear y Mohammad H Askariazad (2012). "Decision support for build-to-order supply chain management through multiobjective optimization". En: *International Journal of Production Economics* 135.1, págs. 24-36 (vid. pág. 51).

Marti, K (1999). "Path planning for robots under stochastic uncertainty". En: *Optimization* 45.1-4, págs. 163-195 (vid. pág. 24).

Marti, Kurt y Shihong Qu (1998). "Path planning for robots by stochastic optimization methods". En: *Journal of Intelligent and robotic Systems* 22.2, págs. 117-127 (vid. pág. 24).

Masehian, Ellips y Hossein Kakahaji (2014). "NRR: a nonholonomic random replanner for navigation of car-like robots in unknown environments". En: *Robotica* 32.7, págs. 1101-1123 (vid. pág. 151).

Masehian, Ellips y Davoud Sedighizadeh (2010). "Multi-objective PSO-and NPSO-based algorithms for robot path planning". En: *Advances in electrical and computer engineering* 10.4, págs. 69-76 (vid. pág. 24).

Mattson, Christopher A y Achille Messac (2005). "Pareto frontier based concept selection under uncertainty, with visualization". En: *Optimization and Engineering* 6.1, págs. 85-115 (vid. págs. 50, 56, 57).

Maturana, Daniel y Sebastian Scherer (mayo de 2015). "3D Convolutional Neural Networks for landing zone detection from LiDAR". En: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2015-June. June. IEEE, págs. 3471-3478. ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139679 (vid. págs. 2, 96, 97).

Maza, Iván y col. (ene. de 2011). "Experimental Results in Multi-UAV Coordination for Disaster Management and Civil Security Applications". En: *Journal of Intelligent & Robotic Systems* 61.1-4, págs. 563-585. ISSN: 0921-0296. DOI: 10.1007/s10846-010-9497-5 (vid. pág. 78).

McLain, Timothy W y Randal W Beard (2005). "Coordination variables, coordination functions, and cooperative timing missions". En: *Journal of Guidance, Control, and Dynamics* 28.1, págs. 150-161 (vid. pág. 2).

McLain, Timothy W, Phillip R Chandler y Meir Pachter (2000). "A decomposition strategy for optimal coordination of unmanned air vehicles". En: *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*. Vol. 1. 6. IEEE, págs. 369-373 (vid. pág. 2).

Mclnnes, CR (1998). "Satellite attitude slew manoeuvres using inverse control". En: *The Aeronautical Journal* 102.1015, págs. 259-266 (vid. pág. 175).

Mekonnen, Gossaye, Sanjeev Kumar y PM Pathak (2016). "Wireless hybrid visual servoing of omnidirectional wheeled mobile robots". En: *Robotics and Autonomous Systems* 75, págs. 450-462. DOI: 10.1016/j.robot.2015.08.008 (vid. pág. 149).

Melchior, Nik A y Reid Simmons (2007). "Particle RRT for path planning with uncertainty". En: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, págs. 1617-1624 (vid. pág. 124).

Meng, AC-C (1988). "Free space modeling and geometric motion planning under unexpected obstacles". En: *Proceedings. The Fourth Conference on Artificial Intelligence Applications*. IEEE Computer Society, págs. 82-83 (vid. pág. 27).

Metaxiotis, Konstantinos y Konstantinos Liagkouras (2012). "Multiobjective evolutionary algorithms for portfolio management: A comprehensive literature review". En: *Expert Systems with Applications* 39.14, págs. 11685-11698 (vid. pág. 51).

Miettinen, KM (1999). "Nonlinear Multiobjective Optimization Kluwer Academic Publishers". En: *Boston, Massachusetts* (vid. pág. 51).

Miller, Rupert y David Siegmund (1982). "Maximally selected chi square statistics". En: *Biometrics*, págs. 1011-1016 (vid. pág. 22).

Min, Chohong y Frédéric Gibou (2006). "A second order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids". En: *Journal of Computational Physics* 219.2, págs. 912-929. ISSN: 00219991. DOI: 10.1016/j.jcp.2006.07.019 (vid. pág. 98).

Mohamed, Sherif AS y col. (2019). "A Survey on Odometry for Autonomous Navigation Systems". En: *IEEE Access* 7, págs. 97466-97486 (vid. pág. 28).

Mora, Marta C y Josep Tornero (2008). "Path planning and trajectory generation using multi-rate predictive artificial potential fields". En: *2008 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, págs. 2990-2995 (vid. pág. 27).

Morales, Yoichi y col. (2013). "Human-comfortable navigation for an autonomous robotic wheelchair". En: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, págs. 2737-2743. DOI: 10.1109/IROS.2013.6696743 (vid. pág. 148).

Münzer, Stefan y col. (2006). "Computer-assisted navigation and the acquisition of route and survey knowledge". En: *Journal of environmental psychology* 26.4, págs. 300-308. DOI: 10.1016/j.jenvp.2006.08.001 (vid. pág. 148).

Nagy, Bryan y Alonzo Kelly (2001). "Trajectory generation for car-like robots using cubic curvature polynomials". En: *Field and Service Robots* 11 (vid. pág. 42).

Navarro-Perez, Alvaro y Jaime Buitrago (2016). "An extended RRT approach for car-like systems". En: *2016 IEEE Colombian Conference on Robotics and Automation (CCRA)*. IEEE, págs. 1-6 (vid. pág. 24).

Nazif, Ali Nasri, Ehsan Iranmanesh y Ali Mohades (2008). "Multiple Robots Tasks Allocation: An Auction-Based Approach Using Dynamic-Domain RRT". En: *Computer Society of Iran Computer Conference*. Springer, págs. 795-798 (vid. pág. 24).

Nevalainen, Olli y col. (2017). "Individual tree detection and classification with UAV-Based photogrammetric point clouds and hyperspectral imaging". En: *Remote Sensing* 9.3. ISSN: 20724292. DOI: 10.3390/rs9030185 (vid. pág. 2).

Newcome, Laurence R (2004). *Unmanned aviation: a brief history of unmanned aerial vehicles*. American Institute of Aeronautics y Astronautics (vid. pág. 64).

Nieuwenhuisen, Matthias y Sven Behnke (2014). "Hierarchical Planning with 3D Local Multiresolution Obstacle Avoidance for Micro Aerial Vehicles". En: *Joint 45th International Symposium on Robotics (ISR) and 8th German Conference on Robotics (ROBOTIK)* June, págs. 598-604 (vid. pág. 79).

Niu, Lei y Guobin Zhuo (2008). "An Improved Real 3D a* Algorithm for Difficult Path Finding Situation". En: *Proceeding of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 37, págs. 927-930 (vid. pág. 96).

Nonami, Kenzo y col. (2010). *Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles*. Springer Science & Business Media (vid. pág. 63).

Norbert Tränapp, IABG (2001). "Preliminary study on integration of unmanned aerial vehicles into future air traffic management". En: *CARE Innovative Action, Tech. Rep* (vid. pág. 68).

*Normativa Sobre Drones en España [2019] - Aerial Insights* (s.f.). `https://www.aerial-insights.co/blog/normativa-drones-espana/` (accessed on 28 December 2018) (vid. pág. 116).

Nosrati, Masoud, Ronak Karimi Hojat Allah Hasanvand e including D Original (2012). "Investigation of the * (Star) Search Algorithms: Characteristics, Methods and Approaches". En: *World Applied Programming* 2.4, págs. 251-256. ISSN: 2222-2510 (vid. pág. 96).

Ollero, A y O Amidi (1991). "Predictive path tracking of mobile robots. Application to the CMU Navlab". En: *Proceedings of 5th International Conference on Advanced Robotics, Robots in Unstructured Environments, ICAR*. Vol. 91, págs. 1081-1086 (vid. pág. 46).

Ollero Baturone, Aníbal (2001). "Robótica: manipuladores y robots móviles". En: (vid. pág. 45).

Ortiz-Arroyo, Daniel (nov. de 2015a). "A hybrid 3D path planning method for UAVs". En: *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE, págs. 123-132. ISBN: 978-1-5090-1784-3. DOI: `10.1109/RED-UAS.2015.7440999` (vid. págs. 95, 97).

— (2015b). "A hybrid 3D path planning method for UAVs". En: *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE, págs. 123-132 (vid. pág. 150).

Osyczka, Andrzej (1985). "Multicriteria optimization for engineering design". En: *Design optimization*. Elsevier, págs. 193-227 (vid. pág. 53).

Pan, Z. y col. (jun. de 2010). "Recent Progress on Programming Methods for Industrial Robots". En: *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, págs. 1-8 (vid. pág. 13).

Pandey, Anish y Dayal R Parhi (2017). "Optimum path planning of mobile robot in unknown static and dynamic environments using Fuzzy-Wind Driven Optimization algorithm". En: *Defence Technology* 13.1, págs. 47-58. DOI: 10.1016/j.dt.2017.01.001 (vid. pág. 150).

Pantano, C. y col. (2007). "A low numerical dissipation patch-based adaptive mesh refinement method for large-eddy simulation of compressible flows". En: *Journal of Computational Physics* 221.1, págs. 63-87. ISSN: 00219991. DOI: 10.1016/j.jcp.2006.06.011 (vid. pág. 98).

Paranjape, Aditya A y col. (2015). "Motion primitives and 3D path planning for fast flight through a forest". En: *The International Journal of Robotics Research* 34.3, págs. 357-377 (vid. pág. 31).

Pehlivanoglu, Y. Volkan (ene. de 2012). "A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV". En: *Aerospace Science and Technology* 16.1, págs. 47-55. ISSN: 12709638. DOI: 10.1016/j.ast.2011.02.006 (vid. pág. 79).

Pemmaraju, Sriram y Steven Skiena (2003). *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica®*. Cambridge university press (vid. pág. 129).

Peng Cheng, James Keller y Vijay Kumar (sep. de 2008). "Time-optimal UAV trajectory planning for 3D urban structure coverage". En: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, págs. 2750-2757. ISBN: 978-1-4244-2057-5. DOI: 10.1109/IROS.2008.4650988 (vid. pág. 125).

Pepy, Romain, Alain Lambert y Hugues Mounier (2006). "Path Planning using a Dynamic Vehicle Model". En: *2006 2nd International Conference on Information & Communication Technologies*. Vol. 1. IEEE, págs. 781-786. ISBN: 0-7803-9521-2. DOI: 10.1109/ICTTA.2006.1684472 (vid. pág. 151).

Pérez, Lucía Hilario y col. (sep. de 2018). "Path Planning Based on Parametric Curves". En: *Advanced Path Planning for Mobile Entities*. InTech. DOI: 10.5772/intechopen.72574 (vid. pág. 151).

Pérez-rodríguez, Ricardo y Arturo Hernández-aguirre (2017). "Muestreo adaptativo aplicado a la robótica: Revisión del estado de la técnica". En: *Revista*

*Iberoamericana de Automática e Informática Industrial* 14.3, págs. 288-298. ISSN: 1697-7912. DOI: 10.1016/j.riai.2017.05.002 (vid. pág. 85).

Perry, Alexander R (2004). "The flightgear flight simulator". En: *Proceedings of the USENIX Annual Technical Conference* (vid. pág. 143).

Pham, DT y Afshin Ghanbarzadeh (2007). "Multi-objective optimisation using the bees algorithm". En: *3rd International Virtual Conference on Intelligent Production Machines and Systems*, pág. 6 (vid. pág. 51).

Piegl, Les y Wayne Tiller (2012). *The NURBS book*. Springer Science & Business Media (vid. pág. 44).

Pigounakis, Kostis G, Nickolas S Sapidis y Kaklis Panagiotis D (1996). "Fairing spatial B-spline curves". En: *Journal of Ship Research* 40.4, pág. 35 (vid. pág. 151).

Porteous, Ian R (2001). *Geometric differentiation: for the intelligence of curves and surfaces*. Cambridge University Press (vid. pág. 38).

Potvin, Jean-Yves, Ying Xu e Ilham Benyahia (abr. de 2006). "Vehicle routing and scheduling with dynamic travel times". En: *Computers & Operations Research* 33.4, págs. 1129-1137. ISSN: 03050548. DOI: 10.1016/j.cor.2004.09.015 (vid. pág. 78).

Protti, Marco y Riccardo Barzan (2007). *UAV Autonomy - Which level is desirable? - which level is acceptable? Alenia Aeronautica Viewpoint*. Inf. téc. ALENIA AERONAUTICA SPA TORINO (ITALY) (vid. pág. 68).

Pulver, Aaron y Ran Wei (ene. de 2018). "Optimizing the spatial location of medical drones". En: *Applied Geography* 90.July 2016, págs. 9-16. ISSN: 01436228. DOI: 10.1016/j.apgeog.2017.11.009 (vid. pág. 94).

Qi, Xue y Zhi-jun Cai (2018). "Three-dimensional formation control based on nonlinear small gain method for multiple underactuated underwater vehicles". En: *Ocean Engineering* 151, págs. 105-114. DOI: 10.1016/j.oceaneng.2018.01.032 (vid. pág. 149).

Qu, Yaohong, Yintao Zhang y Youmin Zhang (mayo de 2014). "Optimal flight path planning for UAVs in 3-D threat environment". En: *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, págs. 149-155. ISBN: 978-1-4799-2376-2. DOI: 10.1109/ICUAS.2014.6842250 (vid. págs. 95, 97, 150).

Raja, P y S Pugazhenthi (2009). "Path planning for mobile robots in dynamic environments using particle swarm optimization". En: *2009 International Conference on Advances in Recent Technologies in Communication and Computing*. IEEE, págs. 401-405 (vid. pág. 24).

Ramasamy, Subramanian y col. (2016). "LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid". En: *Aerospace Science and Technology* 55, págs. 344-358. DOI: 10.1016/j.ast.2016.05.020 (vid. pág. 149).

Ravankar, Abhijeet, Ankit A Ravankar y col. (2016). "SHP: smooth hypocycloidal paths with collision-free and decoupled multi-robot path planning". En: *International Journal of Advanced Robotic Systems* 13.3, pág. 133 (vid. pág. 41).

Ravankar, Abhijeet, Ankit A. Ravankar y col. (2018). "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges". En: *Sensors (Switzerland)* 18.9, págs. 1-30. ISSN: 14248220. DOI: 10.3390/s18093170 (vid. págs. 40, 42, 45).

Reed, Patrick M y col. (2013). "Evolutionary multiobjective optimization in water resources: The past, present, and future". En: *Advances in water resources* 51, págs. 438-456 (vid. pág. 51).

Reeds, James y Lawrence Shepp (1990). "Optimal paths for a car that goes both forwards and backwards". En: *Pacific journal of mathematics* 145.2, págs. 367-393 (vid. págs. 39, 42).

Reineman, Benjamin D. y col. (jul. de 2013). "Development and Testing of Instrumentation for UAV-Based Flux Measurements within Terrestrial and Marine Atmospheric Boundary Layers". En: *Journal of Atmospheric and Oceanic Technology* 30.7, págs. 1295-1319. ISSN: 0739-0572. DOI: 10.1175/JTECH-D-12-00176.1 (vid. pág. 94).

Ren, Xiaoping y Zixing Cai (2010). "Kinematics model of unmanned driving vehicle". En: *2010 8th World Congress on Intelligent Control and Automation*. IEEE, págs. 5910-5914. DOI: 10.1109/WCICA.2010.5554512 (vid. pág. 148).

Reyes-Sierra, Margarita, CA Coello Coello y col. (2006). "Multi-objective particle swarm optimizers: A survey of the state-of-the-art". En: *International journal of computational intelligence research* 2.3, págs. 287-308 (vid. pág. 51).

Rivera, Diego Mauricio, Flavio Augusto Prieto y Ricardo Ramirez (oct. de 2012). "Trajectory Planning for UAVs in 3D Environments Using a Moving Band in Potential Sigmoid Fields". En: *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*. IEEE, págs. 115-119. ISBN: 978-0-7695-4906-4. DOI: 10.1109/SBR-LARS.2012.26 (vid. págs. 95, 97, 150).

Roberge, Vincent, Mohammed Tarbouchi y Gilles Labonté (2012). "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning". En: *IEEE Transactions on industrial informatics* 9.1, págs. 132-141 (vid. pág. 24).

Rodríguez, Rosa M y col. (2013). "Autonomous Management of an UAV Airfield". En: *Proceedings of the 3rd International Conference on Application and Theory of Automation in Command and Control Systems, Naples, Italy*, págs. 28-30 (vid. pág. 94).

Rodríguez-Seda, Erick J (2014). "Decentralized trajectory tracking with collision avoidance control for teams of unmanned vehicles with constant speed". En: *2014 American Control Conference*. IEEE, págs. 1216-1223. DOI: 10.1109/ACC.2014.6859184 (vid. pág. 148).

Runge, Carl (1901). "Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten". En: *Zeitschrift für Mathematik und Physik* 46.224-243, pág. 20 (vid. pág. 131).

Ryde, Julian y Huosheng Hu (2010). "3D mapping with multi-resolution occupied voxel lists". En: *Autonomous Robots* 28.2, pág. 169 (vid. pág. 98).

Sahingoz, Ozgur Koray (abr. de 2014). "Generation of Bezier Curve-Based Flyable Trajectories for Multi-UAV Systems with Parallel Genetic Algo-

rithm". En: *Journal of Intelligent & Robotic Systems* 74.1-2, págs. 499-511. ISSN: 0921-0296. DOI: 10.1007/s10846-013-9968-6 (vid. pág. 79).

Salichs, Miguel A y Maria Malfaz (2011). "A new approach to modeling emotions and their use on a decision-making system for artificial agents". En: *IEEE Transactions on affective computing* 3.1, págs. 56-68 (vid. pág. 12).

Salichs, Miguel A. y col. (2006). "Maggie: A robotic platform for human-robot social interaction". En: *2006 IEEE Conference on Robotics, Automation and Mechatronics* July. DOI: 10.1109/RAMECH.2006.252754 (vid. pág. 11).

Samaniego, Franklin, Javier Sanchis, Sergio Garcia-Nieto y col. (oct. de 2017). "UAV motion planning and obstacle avoidance based on adaptive 3D cell decomposition: Continuous space vs discrete space". En: *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*. IEEE, págs. 1-6. ISBN: 978-1-5386-3894-1. DOI: 10.1109/ETCM.2017.8247533 (vid. págs. 100, 129).

— (2018). "Comparative Study of 3-Dimensional Path Planning Methods Constrained by the Maneuverability of Unmanned Aerial Vehicles". En: *2018 7th International Conference on Systems and Control, ICSC 2018* 1, págs. 13-20. DOI: 10.1109/ICoSC.2018.8587810 (vid. págs. 121, 150).

— (2020). "Smooth 3D Path Planning by Means of Multiobjective Optimization for Fixed-Wing UAVs". En: *Electronics* 9.1, pág. 51 (vid. págs. 33, 46, 47, 59, 76).

Samaniego, Franklin, Javier Sanchis, Sergio García-Nieto y Raúl Simarro (2019). "Recursive Rewarding Modified Adaptive Cell Decomposition (RR-MACD): A Dynamic Path Planning Algorithm for UAVs". En: *Electronics* 8.3, pág. 306 (vid. págs. 33, 167, 168).

Samaniego, Franklin, Javier Sanchis, Sergio García-Nieto y Raul Simarro (2017). "UAV motion planning and obstacle avoidance based on adaptive 3D cell decomposition: Continuous space vs discrete space". En: *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*. IEEE, págs. 1-6 (vid. pág. 33).

Samaniego, Franklin, Javier Sanchís y col. (2018). "Comparative study of 3-dimensional path planning methods constrained by the maneuverability of unmanned aerial vehicles". En: *2018 7th International Conference on Systems and Control (ICSC)*. IEEE, págs. 13-20 (vid. págs. 33, 46, 76).

Samet, H y A Kochut (2002). "Octree approximation an compression methods". En: *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, págs. 460-469. DOI: 10.1109/TDPVT.2002.1024101 (vid. págs. 83, 124).

Samet, Hanan y Andrzej Kochut (2002). "Octree approximation an compression methods". En: *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*. Citeseer, págs. 460-469 (vid. pág. 100).

Sayadi, Fares y col. (2009). "Multi-objective optimization using the bees Algorithm in time-varying channel for MIMO MC-CDMA systems". En: *European Journal of Scientific Research* 33.3, págs. 411-428 (vid. pág. 51).

Schoenberg, Isaac J (1988). "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions". En: *I. J. Schoenberg Selected Papers*. Boston, MA: Birkhäuser Boston, págs. 58-87. DOI: 10.1007/978-1-4899-0433-1_2 (vid. págs. 125, 131).

Schoenberg, Isaac Jacob (1946). "Contributions to the problem of approximation of equidistant data by analytic functions. Part B. On the problem of osculatory interpolation. A second class of analytic approximation formulae". En: *Quarterly of Applied Mathematics* 4.2, págs. 112-141 (vid. pág. 131).

Schouwenaars, Tom, Jonathan How y Eric Feron (2004). "Receding horizon path planning with implicit safety guarantees". En: *Proceedings of the 2004 American control conference*. Vol. 6. IEEE, págs. 5576-5581 (vid. pág. 38).

Segovia, A y col. (1991). "Comparative study of the different methods of path generation for a mobile robot in a free environment". En: *Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments*. IEEE, págs. 1667-1670 (vid. pág. 42).

Shin, Dong Hun, Sanjiv Singh y W Whittaker (1990). *Path generation for robot vehicles using composite clothoid segments, The Robotics Institute*. Inf. téc. Internal Report CMU-RI-TR-90-31, Carnegie-Mellon University (vid. pág. 41).

— (1992). "Path generation for a robot vehicle using composite clothoid segments". En: *IFAC Proceedings Volumes* 25.6, págs. 443-448 (vid. pág. 39).

Siciliano, Bruno y Oussama Khatib (2016). *Springer handbook of robotics*. Springer (vid. pág. 62).

Silvério, Joao y col. (2015). "Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems". En: *2015 IEEE / RSJ international conference on intelligent robots and systems (IROS)*. IEEE, págs. 464-470 (vid. pág. 11).

Skoldstam, M, Knut Akesson y Martin Fabian (2007). "Modeling of discrete event systems using finite automata with variables". En: *Decision and Control, 2007 ...*, págs. 3387-3392. DOI: 10.1109/CDC.2007.4434894 (vid. pág. 105).

Soueres, Philippe y J-P Laumond (1996). "Shortest paths synthesis for a car-like robot". En: *IEEE Transactions on Automatic Control* 41.5, págs. 672-688 (vid. pág. 41).

Steffen, Martin y Gianluigi Zavattaro (2012). *Lecture Notes in Computer Science: Preface*. Vol. 3535. ISBN: 9783642334146 (vid. pág. 13).

Stentz, Anthony (1994). "Optimal and Efficient Path Planning for Partially-Known Environments". En: *In ICRA* 94, págs. 3310-3317 (vid. pág. 96).

— (1995). "Optimal and Efficient Path Planning for Unknown and Dynamic Environments". En: *International Journal of Robotics and Automation* 10, págs. 89-100 (vid. pág. 87).

Stuchlík, Radim y col. (2015). "Unmanned Aerial Vehicle–Efficient mapping tool available for recent research in polar regions". En: *Czech Polar Reports* 5.2, págs. 210-221. ISSN: 18050689. DOI: 10.5817/CPR2015-2-18 (vid. pág. 94).

Sukharev, Aleksandr G (1971). "Optimal strategies of the search for an extremum". En: *USSR Computational Mathematics and Mathematical Physics* 11.4, págs. 119-137 (vid. pág. 25).

Sun, Yi y col. (2011). "Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects". En: *The International Journal of Advanced Manufacturing Technology* 55.5-8, págs. 723-739 (vid. pág. 51).

Szirmay-Kalos, L. y G. Márton (jun. de 1998). "Worst-case versus average case complexity of ray-shooting". En: *Computing* 61.2, págs. 103-131. ISSN: 0010-485X. DOI: 10.1007/BF02684409 (vid. pág. 96).

Talbi, El-Ghazali (2009). *Metaheuristics: from design to implementation*. Vol. 74. John Wiley & Sons (vid. pág. 52).

Thanou, Michalis y Anthony Tzes (mayo de 2014). "Distributed visibility-based coverage using a swarm of UAVs in known 3D-terrains". En: *2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP)*. IEEE, págs. 425-428. ISBN: 978-1-4799-2890-3. DOI: 10.1109/ISCCSP.2014.6877904 (vid. págs. 95, 97, 150).

Tompkins, P., A. Stentz y D. Wettergreen (mar. de 2004). "Global path planning for Mars rover exploration". En: *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*. Vol. 2, 801-815 Vol.2. DOI: 10.1109/AERO.2004.1367681 (vid. pág. 13).

Tong, Han y col. (2012). "Path planning of UAV based on Voronoi diagram and DPSO". En: *Procedia Engineering* 29, págs. 4198-4203 (vid. pág. 27).

Toponogov, Victor A (2006). *Differential geometry of curves and surfaces*. Springer (vid. pág. 127).

Trisna, Trisna y col. (2016). "Multi-objective optimization for supply chain management problem: A literature review". En: *Decision Science Letters* 5.2, págs. 283-316 (vid. pág. 51).

Tsai, Ching-Chih, Hsu-Chih Huang y Cheng-Kai Chan (2011). "Parallel elite genetic algorithm and its application to global path planning for autono-

mous robot navigation". En: *IEEE Transactions on Industrial Electronics* 58.10, págs. 4813-4821 (vid. pág. 24).

Tuncer, Adem y Mehmet Yildirim (2012). "Dynamic path planning of mobile robots with improved genetic algorithm". En: *Computers & Electrical Engineering* 38.6, págs. 1564-1572 (vid. pág. 24).

*UNMANNED APPLIED SOLUTIONS* (2018). `https://www.microdrones.com/en/industry-experts/` (vid. pág. 94).

Vadakkepat, Prahlad, Kay Chen Tan y Wang Ming-Liang (2000). "Evolutionary artificial potential fields and their application in real time robot path planning". En: *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*. Vol. 1. IEEE, págs. 256-263 (vid. pág. 27).

Valavanis, Kimon P y George J Vachtsevanos (2015). *Handbook of Unmanned Aerial Vehicles*. Ed. por Kimon P. Valavanis y George J. Vachtsevanos. Dordrecht: Springer Netherlands, págs. 2993-3009. ISBN: 978-90-481-9706-4. DOI: `10.1007/978-90-481-9707-1` (vid. pág. 94).

Vanegas, Gloria y col. (oct. de 2018). "Smooth 3D path planning for non-holonomic UAVs". En: *2018 7th International Conference on Systems and Control (ICSC)*. 1. IEEE, págs. 1-6. ISBN: 978-1-5386-8537-2. DOI: `10.1109/ICoSC.2018.8587835` (vid. págs. 121, 150).

Velasco, J y Sergio García-Nieto (2014). "Unmanned aerial vehicles model identification using multi-objective optimization techniques". En: *IFAC Proceedings Volumes* 47.3, págs. 8837-8842 (vid. págs. 71, 73).

Velasco Carrau, Jesús (2014). "Identificación de modelos dinámicos y ajuste de controladores basado en algoritmos evolutivos multiobjetivo". En: (vid. pág. 71).

Velasco Carrau, Jesús y col. (2012). "Desarrollo y evaluación de una estación de control de tierra para vehículos aéreos no tripulados". En: *Actas de las XXXIII Jornadas de Automática* (vid. pág. 71).

Velasco-Carrau, J. y col. (feb. de 2016). "Multi-Objective Optimization for Wind Estimation and Aircraft Model Identification". En: *Journal of Guidance, Control, and Dynamics* 39.2, págs. 372-389. ISSN: 0731-5090. DOI: 10.2514/1.G001294 (vid. págs. 121, 127, 155, 168).

Verscheure, L. y col. (ago. de 2010). "Dijkstra's algorithm applied to 3D skeletonization of the brain vascular tree: Evaluation and application to symbolic". En: *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE, págs. 3081-3084. ISBN: 978-1-4244-4123-5. DOI: 10.1109/IEMBS.2010.5626112 (vid. pág. 96).

Vershik, AM y LD Faddeev (1995). "Lagrangian mechanics in invariant form". En: *40 Years In Mathematical Physics*. World Scientific, págs. 427-438 (vid. pág. 19).

Wallace, Richard S y col. (1985). "First Results in Robot Road-Following." En: *IJCAI*. Citeseer, págs. 1089-1095 (vid. pág. 45).

Wang, Gai-Ge, Haicheng Eric Chu y Seyedali Mirjalili (feb. de 2016). "Three-dimensional path planning for UCAV using an improved bat algorithm". En: *Aerospace Science and Technology* 49, págs. 231-238. ISSN: 12709638. DOI: 10.1016/j.ast.2015.11.040 (vid. págs. 96, 97).

Warner, Frank W (2013). *Foundations of differentiable manifolds and Lie groups*. Vol. 94. Springer Science & Business Media (vid. pág. 38).

Weisstein, Eric W (2009). "Euler angles". En: (vid. pág. 126).

— (2016a). *Cornu spiral from mathworld–a wolfram web resource. 2016* (vid. pág. 42).

— (2016b). "Fresnel integrals from math-world–a wolfram web resource. 2016". En: *URL http://mathworld. wolfram. com/FresnelIntegrals. html.(date last viewed 6/12/12)* (vid. pág. 42).

Xu, Jian, Man Wang y Lei Qiao (2015). "Dynamical sliding mode control for the trajectory tracking of underactuated unmanned underwater vehicles". En: *Ocean engineering* 105, págs. 54-63. DOI: 10.1016/j.oceaneng.2015.06.022 (vid. pág. 149).

Xue, Feng, Arthur C Sanderson y Robert J Graves (2005). "Multi-objective differential evolution-algorithm, convergence analysis, and applications". En: *2005 IEEE congress on v evolutionary computation*. Vol. 1. IEEE, págs. 743-750 (vid. pág. 51).

Yamaguchi, Fujio (2012). *Curves and surfaces in computer aided geometric design*. Springer Science & Business Media (vid. pág. 132).

Yan, Fei, Yi-Sha Liu y Ji-Zhong Xiao (dic. de 2013). "Path Planning in Complex 3D Environments Using a Probabilistic Roadmap Method". En: *International Journal of Automation and Computing* 10.6, págs. 525-533. ISSN: 1476-8186. DOI: 10.1007/s11633-013-0750-9 (vid. págs. 95, 97, 150).

Yang, Yi-Hua E. y Viktor K. Prasanna (abr. de 2011). "Space-time tradeoff in regular expression matching with semi-deterministic finite automata". En: *2011 Proceedings IEEE INFOCOM*. IEEE, págs. 1853-1861. ISBN: 978-1-4244-9919-9. DOI: 10.1109/INFCOM.2011.5934986 (vid. pág. 105).

Yang, Kwangjin y Salah Sukkarieh (2010). "An analytical continuous-curvature path-smoothing algorithm". En: *IEEE Transactions on Robotics* 26.3, págs. 561-568 (vid. pág. 32).

Yao, Peng, Honglun Wang y Zikang Su (nov. de 2015). "Hybrid UAV path planning based on interfered fluid dynamical system and improved RRT". En: *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, págs. 000829-000834. ISBN: 978-1-4799-1762-4. DOI: 10.1109/IECON.2015.7392202 (vid. págs. 95, 97, 149).

Yaochu Jin y Berhard Sendhoff (mayo de 2008). "Pareto-Based Multiobjective Machine Learning: An Overview and Case Studies". En: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.3, págs. 397-415. DOI: 10.1109/TSMCC.2008.919172 (vid. pág. 152).

Yeh, Hsin-Yi y col. (oct. de 2012). "UOBPRM: A uniformly distributed obstacle-based PRM". En: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, págs. 2655-2662. ISBN: 978-1-4673-1736-8. DOI: 10.1109/IROS.2012.6385875 (vid. págs. 95, 97, 150).

YongBo, Chen y col. (nov. de 2017). "Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm". En: *Neurocomputing* 266, págs. 445-457. ISSN: 09252312. DOI: 10.1016/j.neucom.2017.05.059 (vid. págs. 96, 97).

Yongqiang, Wang y col. (2009). "Fault detection of networked control systems with packet based periodic communication". En: *International Journal of Adaptive Control and Signal Processing* 23.8, págs. 682-698. DOI: 10.1002/acs (vid. pág. 2).

Young, Andrew T (2006). "Understanding astronomical refraction". En: *The Observatory* 126, págs. 82-115 (vid. pág. 126).

Zaloga, Steven J (2011). *Unmanned aerial vehicles: robotic air warfare 1917–2007*. Bloomsbury Publishing (vid. pág. 64).

Zecca, Massimiliano y col. (2008). "Design of the humanoid robot KOBIAN-preliminary analysis of facial and whole body emotion expression capabilities". En: *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, págs. 487-492 (vid. pág. 11).

Zhang, Yudong, Lenan Wu y Shuihua Wang (2013). "UCAV Path Planning by Fitness-Scaling Adaptive Chaotic Particle Swarm Optimization". En: *Mathematical Problems in Engineering* 2013.I, págs. 1-9. ISSN: 1024-123X. DOI: 10.1155/2013/705238 (vid. págs. 96, 97).

Zhu, Lihua, Xianghong Cheng y Fuh-Gwo Yuan (2016). "A 3D collision avoidance strategy for UAV with physical constraints". En: *Measurement* 77, págs. 40-49. DOI: 10.1016/j.measurement.2015.09.006 (vid. pág. 149).