

Grado en Ingeniería Informática

2021-2022

Trabajo Fin de Grado

“MONITORIZACIÓN DE CARRETERAS DESDE UAVS SIMULADOS”

Diego Fernández Plaza

Tutor

Daniel Amigo Herrero

Leganés, Comunidad de Madrid, España, Septiembre 2022



Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento
– No Comercial – Sin Obra Derivada**

Dedicatoria

Quiero dedicar este documento y agradecer por tanto no solo el TFG, si no toda la carrera, a mis padres por darme apoyo durante todo el camino, a mis amigos que han hecho el proceso más ameno y me han soportado hablar de él constantemente y por último a mi tutor Daniel que me ha brindado ayuda en todo el proceso.

“Viaje antes que destino”

Resumen

Los drones pueden capturar información desde perspectivas a las que un ser humano no puede llegar, e incluso procesarla para obtener conocimiento de forma rápida y precisa. Uno de sus usos más comunes son las tareas de monitorización de carreteras. En estas tareas actualmente se utiliza el dron como los ojos, siendo un humano quien realiza la inferencia del conocimiento de lo que sucede en la carretera.

Este trabajo pretende abarcar tanto la captura de la información como su posterior explotación de forma automática, haciendo una misión autónoma capaz de detectar los vehículos e infracciones que estos realicen. Para ello se estudian distintas soluciones y algoritmos existentes. En concreto, la finalidad del trabajo es la de poder estimar la velocidad de los vehículos en un tramo de la carretera mediante la grabación de este y la utilización de la visión por computador haciendo uso de distintos algoritmos mediante la herramienta OpenCV.

Para aplicar estos algoritmos, primero se necesita un vídeo que se consigue mediante la cámara del dron. Debido a las imposibilidades de realizar las pruebas de este proyecto en un entorno real, se necesita de un simulador de drones para la realización de este y se ha optado por AirSim mediante Unreal Engine y para aportar más realismo se usará la extensión Cesium, que aporta un mapa del mundo que utiliza una visión geográfica del mismo.

Como el trabajo consiste en la estimación de la velocidad de los vehículos que recorren un tramo de una carretera, los vehículos dentro de la simulación seguirán un camino establecido para así poder realizar una simulación de un desplazamiento en ese tramo a una velocidad que se pondrá como límite, además recogerán información importante para su posterior análisis como velocidad de los mismo, posición respecto al dron, posición del dron...

Entonces, utilizando MavSDK y MAVLink para conectar el dron al computador y moverle y mediante el software del controlador de vuelo PX4 se es capaz de realizar una misión autónoma que vuele a un tramo de la carretera y comience la grabación ya comentada para posteriormente procesar el vídeo resultante.

Palabras clave: Dron, OpenCV, AirSim, Unreal Engine, Cesium, MavSDK, MAVLink, PX4.

Abstract

Drones can provide information that a human cannot see and process it much faster. This is especially useful for road monitoring tasks and to be able to detect vehicles on the roads and the infringements they make.

By studying different existing solutions and algorithms, as well as analysing how a drone works, it has been possible to get an idea of how to approach the project.

The aim of the work is to be able to estimate the speed of vehicles on a stretch of road by recording it and using computer vision, making use of different algorithms using the OpenCV tool.

To apply these algorithms, a video is first needed, which is obtained by means of the drone's camera. Due to the impossibility of testing this project in a real environment, a drone simulator is needed to carry it out, and AirSim using Unreal Engine has been chosen. To provide more realism, the Cesium extension will be used, which provides a map of the world using a geographical view of it.

As the work consists of estimating the speed of vehicles travelling along a stretch of a road, the vehicles within the simulation will follow an established path in order to simulate a displacement along that stretch at a speed that will be set as a limit, and will also collect important information for later analysis, such as their speed, position with respect to the drone, position of the drone, etc.

Then, using MavSDK and MAVLink to connect the drone to the computer and move it and using the PX4 flight controller software, we are able to carry out an autonomous mission that flies to a section of the road and starts the aforementioned recording to later process the resulting video.

Keywords: Drone, OpenCV, AirSim, Unreal Engine, Cesium, MavSDK, MAVLink, PX4.

Tabla de contenidos

Dedicatoria.....	ii
Resumen	iv
Abstract.....	v
Índice de tablas	xi
Índice de figuras	xiii
1 INTRODUCCIÓN	1
1.1 Motivación	1
1.2 Objetivos	3
1.3 Estructura del documento	4
2 ESTADO DEL ARTE.....	6
2.1 Dron	6
2.1.1 Movimiento del dron	7
2.1.2 Clasificaciones de dron.....	10
2.1.3 Enjambre de drones	11
2.1.4 Sensores	12
2.1.5 Formas de uso	15
2.2 Controlador de vuelo	15
2.2.1 PX4	16
2.2.2 Diferencias entre Ardupilot y PX4	17
2.2.3 Conclusión de uso de PX4.....	17
2.3 MAVLink.....	18
2.4 MavSDK	19
2.5 Simulación de drones	19
2.5.1 AirSim	19
2.5.2 Comparación Airsim y otros simuladores	20
2.5.3 Unreal Engine	21
2.5.4 Cesium	22
2.6 Lenguaje de programación.....	23
2.6.1 Python.....	23
2.6.2 OpenCV	24
2.6.3 Blueprints de Unreal Engine.....	24
2.7 Inteligencia Artificial	25
2.8 Computer Vision.....	25

2.8.1	Detección de vehículos	26
2.8.2	Tracking de un objeto	27
2.8.3	Estimación de la velocidad	27
2.8.4	Conceptos importantes del sistema.....	28
2.9	Soluciones existentes	32
3	PLANIFICACIÓN Y PRESUPUESTO.....	34
3.1	Metodología de trabajo	34
3.2	Planificación del proyecto	35
3.2.1	Explicación fases del proyecto	35
3.2.2	Planificación inicial del proyecto	36
3.2.3	Planificación real del proyecto	37
3.3	Presupuesto	38
3.3.1	Costes materiales	38
3.3.2	Costes personales.....	39
3.3.3	Costes totales	39
4	MARCO REGULADOR.....	41
4.1	Legislación española.....	41
4.2	Legislación europea	42
4.2.1	Categoría abierta.....	42
4.2.2	Categoría específica.....	43
4.2.3	Categoría certificada.....	43
4.3	Ley Orgánica de Protección de Datos.....	43
4.4	Multas por Inteligencia Artificial	44
5	ENTORNO SOCIO-ECONÓMICO	45
5.1	Impacto Social	45
5.2	Impacto Económico	46
5.3	Impacto Medioambiental	46
6	DISEÑO DE LA SOLUCIÓN	47
6.1	Tecnologías empleadas	47
6.2	Identificación de requisitos	48
6.3	Casos de uso.....	49
6.3.1	Matriz de trazabilidad.....	51
6.4	Arquitectura del sistema	53

6.4.1	Conexión del sistema e inicialización del mismo.....	54
6.5	Diseño de la solución	55
6.5.1	Diseño de la misión del dron	55
6.5.2	Diseño de las tareas de los coches	56
6.5.3	Diseño del algoritmo de estimación de velocidad	59
6.6	Alternativas a la solución / Alcance de la solución	63
7	EVALUACIÓN DE LA SOLUCIÓN	64
7.1	Pruebas de la misión autónoma	65
7.2	Pruebas del seguimiento de un camino por los vehículos	67
7.3	Pruebas del algoritmo de estimación de velocidad	68
7.3.1	Prueba de detección en un entorno real	69
7.3.2	Prueba con un solo vehículo	70
7.3.3	Prueba con dos vehículos	71
7.3.4	Prueba con cuatro vehículos	72
7.3.5	Prueba desde encima de la carretera.....	73
7.3.6	Evaluación final	74
8	CONCLUSIONES Y TRABAJOS FUTUROS	76
8.1	Conclusiones del proyecto	76
8.2	Conclusiones personales	77
8.3	Trabajos futuros	77
9	BIBLIOGRAFÍA.....	79
Anexo A.	Resumen en inglés	84
A.1	Introduction	84
A.2	State of the art.....	85
A.2.1	Drone	85
A.2.2	Simulation.....	86
A.2.3	Computer Vision.....	87
A.3	Socioeconomic and legal environment.....	88
A.3.1	Socioeconomic Enviroment.....	88
A.3.2	Legal enviroment	89
A.4	Solution design	91
A.4.1	Drone mission design.....	91
A.4.2	Design of car tasks	92

A.4.3	Design of the speed estimation algorithm.....	93
A.5	Planning and Budget.....	95
A.5.1	Planning	95
A.5.2	Budget.....	96
A.6	Results	98
A.7	Conclusions	100
A.7.1	Project conclusions	100
A.7.2	Personal conclusions.....	101
Anexo B.	Requisitos	102
B.1	Requisitos funcionales.....	102
B.2	Requisitos no funcionales.....	109

Índice de tablas

Tabla 1. Diagrama de Gantt de la planificación inicial	36
Tabla 2. Planificación inicial del proyecto	37
Tabla 3. Diagrama de Gantt de la planificación real	37
Tabla 4. Planificación real del proyecto	37
Tabla 5. Costes materiales	39
Tabla 6. Costes personales.....	39
Tabla 7. Costes totales	40
Tabla 8. Formato base de los requisitos	48
Tabla 9. Modelo de tabla de los casos de uso.....	49
Tabla 10. Caso de uso 01	50
Tabla 11. Caso de uso 02.....	50
Tabla 12. Caso de uso 03.....	51
Tabla 13. Matriz de trazabilidad de los requisitos funcionales vs Casos de uso	52
Tabla 14. Marcas de tiempo prueba de un vehículo.	71
Tabla 15. Marcas de tiempo prueba de dos vehículos.	71
Tabla 16. Marcas de tiempo prueba de cuatro vehículos.	73
Tabla 17. Gantt chart of actual planning	96
Tabla 18. Actual project planning	96
Tabla 19. Material costs.....	97
Tabla 20. Personal costs	97
Tabla 21. Total costs.....	98
Tabla 22. Requisito funcional 01.....	102
Tabla 23. Requisito funcional 02.....	102
Tabla 24. Requisito funcional 03.....	102
Tabla 25. Requisito funcional 04.....	103
Tabla 26. Requisito funcional 05.....	103
Tabla 27. Requisito funcional 06.....	103
Tabla 28. Requisito funcional 07.....	104
Tabla 29. Requisito funcional 08.....	104
Tabla 30. Requisito funcional 09.....	104
Tabla 31. Requisito funcional 10.....	105
Tabla 32. Requisito funcional 11.....	105
Tabla 33. Requisito funcional 12.....	105
Tabla 34. Requisito funcional 13.....	106
Tabla 35. Requisito funcional 14.....	106
Tabla 36. Requisito funcional 15.....	106
Tabla 37. Requisito funcional 16.....	107
Tabla 38. Requisito funcional 17.....	107
Tabla 39. Requisito funcional 18.....	107
Tabla 40. Requisito funcional 19.....	108
Tabla 41. Requisito funcional 20.....	108
Tabla 42. Requisito funcional 21.....	108
Tabla 43. Requisito no funcional 01	109

Tabla 44. Requisito no funcional 02.....	109
Tabla 45. Requisito no funcional 03.....	109
Tabla 46. Requisito no funcional 04.....	110
Tabla 47. Requisito no funcional 05.....	110
Tabla 48. Requisito no funcional 06.....	110
Tabla 49. Requisito no funcional 07.....	111
Tabla 50. Requisito no funcional 08.....	111
Tabla 51. Requisito no funcional 09.....	111
Tabla 52. Requisito no funcional 10.....	112
Tabla 53. Requisito no funcional 11.....	112
Tabla 54. Requisito no funcional 12.....	112
Tabla 55. Requisito no funcional 13.....	113
Tabla 56. Requisito no funcional 14.....	113

Índice de figuras

Figura 1. Cuadricóptero [5]	6
Figura 2. Movimiento de Guiñada [8]	8
Figura 3. Movimiento de Aladeo [8]	9
Figura 4. Movimiento de Cabeceo [8]	9
Figura 5. Ejemplo de dron de ala fija [9]	10
Figura 6. Enjambre de drones [10]	12
Figura 7. Ejemplo uso LiDAR [13]	13
Figura 8. Ejemplo uso sensor térmico. [14]	14
Figura 9. Comparación cámara RGB con cámara multispectral NDVI	14
Figura 10. Controlador de vuelo Pixhawk [17]	16
Figura 11. Lenguajes de programación para MAVLink [19]	18
Figura 12. Ejemplo AirSim con Unreal [22]	20
Figura 13. Ejemplo de simulación en Gazebo [26]	21
Figura 14. Ejemplo de interfaz de Unreal Engine	22
Figura 15. Ejemplo de un escenario de Cesium [29]	23
Figura 16. Ejemplo de uso de OpenCV con reconocimiento de rasgos faciales [31] ...	24
Figura 17. Ejemplo de blueprint	25
Figura 18. Ejemplo de detección de vehículos [36]	26
Figura 19. Ejemplo de zona donde calcular velocidad en un vídeo [37]	28
Figura 20. Fórmula de la binarización de una imagen [38]	29
Figura 21. Binariación inversa de una imagen [38]	29
Figura 22. Histograma Otsu	30
Figura 23. Comparación imagen binarizada con imagen real.	30
Figura 24. Algoritmo de un sustractor de fondo	31
Figura 25. Sombras en una máscara	32
Figura 26. Uso de Trello en el proyecto	35
Figura 27. Distribución de drones en España [52]	45
Figura 28. Arquitectura del sistema	53
Figura 29. Reglas de entrada y salida	54
Figura 30. Regla UDP	55
Figura 31. Diagrama de flujo misión del dron	56
Figura 32. Ejemplo camino Spline	57
Figura 33. Diagrama de flujo de las tareas de los vehículos	59
Figura 34. Diagrama de flujo detección de los vehículos	60
Figura 35. Diagrama de flujo del tracking de los vehículos	61
Figura 36. Diagrama de flujo de la estimación de la velocidad	62
Figura 37. Ubicación inicial del dron	64
Figura 38. Ubicación destino del dron	65
Figura 39. Ubicación destino dentro del simulador	66
Figura 40. Prueba de cámara del dron	66
Figura 41. Camino de prueba para el seguimiento de los coches	67
Figura 42. Imágenes del seguimiento de los vehículos.	67
Figura 43. Información que proporciona el CSV	68

Figura 44. Distancia estimada utilizada.....	69
Figura 45. Prueba en un entorno real.....	70
Figura 46. Prueba con un solo vehículo	70
Figura 47. Prueba con dos vehículos	71
Figura 48. Prueba con 4 vehículos.....	72
Figura 49. Prueba con la cámara encima de la carretera	74
Figura 50. Gráfico de comparación entre velocidades.	75
Figura 51. Path to follow example	92
Figura 52. Initial drone location	99
Figura 53. Speed comparison chart.	99

1 INTRODUCCIÓN

En este apartado se va a comentar las motivaciones de por qué se ha realizado, y los motivos de por qué se ha hecho de cierta manera. También se hará un breve paso por los objetivos impuestos al inicio del proyecto, además de explicar la estructura del documento para que exista cierta cohesión en el mismo y el lector sepa dónde ubicarse en cada momento.

Es importante recalcar las motivaciones del proyecto, ya que sirven tanto para saber la intención del autor a la hora de realizarlo tanto como una introducción al resto del documento.

1.1 Motivación

El uso de drones para recogida de información, como puede ser mapeo de ciudades desde otras perspectivas, o su uso para tareas más normales como la fase de prueba de Google para utilizar estos para repartos de tiendas o corporativas [1], hace que el empleo esta tecnología se esté convirtiendo en algo cada vez más común.

Los drones son útiles en la actualidad debido al gran alcance que estos poseen y al gran número de usos distintos que ofrecen, pueden ser usados tanto en cine, fotografía, grabación en eventos deportivos, inspección de edificios, mapeo de terreno, labores de búsqueda o regulación de cultivos. Por estos motivos es un tipo de vehículos que están cogiendo fama en los últimos años. También se espera que se desarrollen lo suficiente como para realizar ya no solo tareas de vigilancia de cultivos si no que puedan fumigar o fertilizar los mismo, o también sean capaces de transportar cargas pesadas. Por último, la investigación sobre esta tecnología está en pleno auge de todo tipo, tanto investigación de campo como el estudio en biología de zonas de complicado acceso como Amazonas o simplemente investigación para la realización algoritmos cada vez más complejos. [2]

También podemos observar que una motivación más de usar drones es la posibilidad que nos ofrecen con toda la información que pueden recoger de manera mucho más sencilla que si lo hiciese un humano, por lo que es otro punto a favor a la hora de empezar a usar estos vehículos en más tareas.

Se plantea esta visión para el manejo de los drones como algo innovador, puesto que a pesar de que sí se usa, no es una aplicación muy extendida dentro de los cuerpos de policía del mundo. Esta aplicación de los drones vendría a sustituir los controles de las carreteras mediante helicópteros si se realiza un movimiento del dron en una zona más extensa o si se usa un sensor que detecta la velocidad podrían usarse para sustituir radares.

Por lo tanto, es interesante explorar esta opción para monitorizar carreteras y su uso dentro de los cuerpos de policía podría abaratar costes y el no depender del trabajo humano y las limitaciones de este como el poder detectar comportamientos anómalos en las carreteras a altas horas de la noche, aunque solo sería un complemento a los trabajos típicos de

control de carreteras porque no se puede sustituir de manera completa debido al porcentaje de fallo que puede haber o situaciones que pueden ser interpretables y no requieren una solución absoluta.

Por lo comentado anteriormente, se puede concluir que su uso en horas altas de la noche, en las que desde un helicóptero incluso con un foco es difícil observar lo que ocurre en carretera, podría beneficiar al uso de los drones de forma autónoma sin tener que ser controlados por policías. Además, podría usarse de manera complementaria a los agentes de tráfico, si el dron detecta una infracción le enviaría a la patrulla que se encuentre por la zona un aviso para detener al vehículo infractor o simplemente que el dron cubriese una zona y un grupo de policías cubriese otra dando la posibilidad de cubrir una mayor cantidad de terreno y evitar así gasto de personal en controlar estos drones.

Si nos fijamos únicamente en la motivación final del sistema, es decir, la estimación de la velocidad a través de drones, hay que mencionar que los drones actuales que usa la Dirección General de Tráfico no poseen la capacidad de medir la velocidad de los vehículos y por tanto este proyecto sería una aproximación a una posible solución si se quisiese estimar la velocidad sin tener que instalar radares. Por lo que en un futuro podría seguirse trabajando por este camino para que ya no solo la universidad se beneficie si no también al propio Gobierno de España. [3]

Aunque una simulación realizada únicamente con AirSim podría funcionar correctamente y no presentar ningún problema, le faltaría el realismo que puede aportar un software que funciona como un controlador de vuelo como PX4, haciendo uso del giroscopio y el acelerómetro para calcular el estado del dron mediante giros y que las físicas de este sean lo más realista posibles. Por lo tanto, se ha combinado estos dos softwares para aportar realismo a la simulación.

Cabe destacar el uso de Cesium y el por qué no se ha realizado en un simple entorno creado a partir de Unreal Engine sin tener que ver con el mundo real. Esto es pues, debido a que ya que se pretende que en un futuro pueda usarse en un entorno real para que sea lo más fiel a este se ha decidido realizarlo mediante esta herramienta, además de mejorar este entorno para que en futuros proyectos universitarios se encuentre más avanzado.

La motivación para realizar este trabajo de investigación es la de resolver los problemas expuestos anteriormente de implementar y mejorar los sistemas de vuelo y reconocimiento de vehículos de forma autónoma. De la misma manera, otro objetivo es mejorar el entorno usado por el Grupo de Inteligencia Artificial Aplicada en el contexto de manejo de drones y su simulación de uso en entornos más realistas mediante Cesium, además de todas las acciones que vienen impuestas por este como el tener coches semiautónomos.

1.2 Objetivos

Una buena definición de objetivos dará como resultado en una óptima forma de realizar el trabajo y en este apartado serán definidos a continuación:

El objetivo principal del trabajo es el de conseguir que el dron sea capaz de reconocer y monitorizar los vehículos que se mueven por una carretera determinada y controlar determinar la velocidad de estos, todo esto mediante un entorno simulados usando Cesium, un plugin de Unreal Engine.

Se usará un entorno simulado debido a la imposibilidad de realizar vuelos reales con drones por la legislación actual, la cual será explicada más adelante. Además, al realizarse desde un entorno simulado hay más tolerancia al fallo, puesto que en caso de error fatal no habrá consecuencias materiales.

Por otro lado, habrá que usar Cesium como se ha comentado anteriormente y colocarse dentro del entorno simulado. Y conseguir que los vehículos dentro de Unreal Engine consigan moverse por el mundo de manera autónoma y que sigan ciertos patrones de comportamiento, usando componentes spline asociados a cada coche para realizar un camino que seguirán, esto será explicado con una mayor extensión en el apartado de desarrollo.

Se podría comentar la adición de carreteras realistas dentro del entorno simulado utilizado como un objetivo del sistema, pero cómo se ha utilizado un sistema anterior para las carreteras simplemente se quedará en una mención.

Por lo tanto, se puede dividir en varios objetivos generales en cuánto al sistema:

- Integración en un entorno simulado dentro de Cesium y UE4.
- Movimiento autónomo de vehículos.
- Sincronización del dron mediante el controlador de vuelo PX4.
- Vuelo del dron a una cierta parte del mapa.
- Monitorización de conducción de vehículos mediante la cámara de AirSim y usando OpenCV.

Ya no centrándose en el sistema si no en los objetivos de este documento y del propio proyecto en sí, podemos destacar el estudio de distintos apartados desde todo lo relativo a los drones como las posibles funcionalidades que estos tienen como la legislación que existe en la actualidad, también el propio análisis de cómo realizar un algoritmo que cumpla con lo dicho o el diseño del sistema en general. Estos requisitos si bien no son del sistema son útiles para la realización de este y el entendimiento para saber que se está haciendo y tener la capacidad de poder explicarlo en este documento.

Para la evaluación del sistema, el objetivo final será el poder comprobar cuanta variación existe entre el resultado obtenido en cuánto a velocidad respecto a la velocidad real obtenida del propio sistema.

Estos son los objetivos necesarios para tener un sistema lo suficientemente funcional para lo comentado anteriormente.

1.3 Estructura del documento

Este documento está dividido en ocho apartados que van a ser explicados brevemente para que cuando se llegue a cada uno de ellos no exista ningún tipo de confusión y se consiga una mayor claridad a lo largo de todo el documento. Se dispone entonces a explicar la estructura del documento de manera ordenada:

- **Introducción:** Como su propio nombre indica es la introducción al resto del documento, donde se pondrá en contexto al lector cuál es el tema del mismo y se comenten las motivaciones del proyecto y sus objetivos. Además, es donde se encuentra este mismo apartado que explica la estructura del documento que se va a leer.
- **Estado del arte:** Se tomarán todos los conceptos utilizados y se explicarán para saber desde que punto se parte a la hora realizar el proyecto. También sirve para explicar conceptos que pueden ser repetidos más adelante y, así, no causar confusión en su uso.
- **Planificación y presupuesto:** En este apartado, se mostrará la planificación que se ha llevado en el proyecto, los cambios que se han realizado en este y la metodología usada para su realización. Realizando también un análisis de presupuesto necesario para este proyecto.
- **Marco regulador:** Como en cualquier proyecto, hay que analizar qué elementos legales pueden inferir y hacer que sean necesarios algún cambio en el mismo.
- **Entorno socio-económico:** Se dispondrá a comentar cualquier elemento del proyecto que pueda tener un impacto en la sociedad en cualquier apartado ya sea en lo económico, social o medioambiental.
- **Diseño de la solución:** Siendo el apartado más extenso del documento, se relatará todo el proceso de desarrollo, pasando por la identificación de todos los requisitos del sistema, las tecnologías utilizadas en este, el desarrollo del mismo, comentando y explicando cómo han sido realizados todos los aspectos del sistema. Para concluir, se comentarán otras posibles soluciones o alternativas a este y que funcionalidades se han probado o descartado por no dar los resultados esperados.
- **Evaluación de la solución:** En este apartado se mostrarán las pruebas que se han realizado para la comprobación de que el sistema funcione correctamente, describiendo la situación inicial de la prueba y qué resultados han dado con una comparación con los resultados reales en cuanto a la velocidad.
- **Conclusiones y trabajos futuros:** Para terminar con el documento se realizará unas conclusiones del mismo, analizando los resultados del sistema y comparando los objetivos iniciales con estos resultados y se relatará que le ha parecido al autor la realización del trabajo. Además, se explicará cómo se podría continuar el sistema para trabajos futuros.
- **Bibliografía:** Es el apartado donde encontramos todas las referencias, webs, e información usada a lo largo de todo el documento.

También cabe recalcar que al final del documento se encuentran dos anexos, uno que corresponde al resumen en inglés necesario según el plan de 2011 del Grado de Ingeniería Informática y otro para los requisitos tanto funcionales como no funcionales del sistema para que no estuviesen presentes en mitad del documento y el lector encontrase varias páginas seguidas de ellos en mitad de la lectura.

2 ESTADO DEL ARTE

En este capítulo, se va a realizar una explicación de cómo se encuentran los elementos que han sido utilizados para la realización del proyecto a fecha de entrega del mismo, describiendo las características de cada uno de ellos y por qué han sido utilizados en el sistema.

Es importante tener una buena base de conocimiento para los conceptos que se van a explicar posteriormente como algoritmos o uso de herramientas de cada uno de los términos descritos en este apartado.

2.1 Dron

Un dron [4] o también llamado en algunos contextos UAV es un vehículo controlado de manera remota que utiliza la fuerza de giro de los motores unidos a sus hélices para elevarse y realizar vuelos. Dependiendo del número de hélices que posea su nombre cambia si tiene cuatro es llamado cuadricóptero, si posee seis es un hexacóptero y si su número de hélices es ocho entonces es un octacóptero.



Figura 1. Cuadricóptero [5]

Las acciones y movimientos de un dron pueden ser realizadas o bien siendo controlados de forma remota o mediante algoritmos y métodos para que estas acciones sean realizadas de manera autónoma. Este trabajo se va a centrar en los drones con vuelos autónomos.

Los drones surgieron en el ámbito militar para realizar misiones de reconocimiento o de ataque al enemigo. Pero en la actualidad están mucho más normalizados en la población siendo utilizados incluso para el ocio. Aunque están muy regulados para que no se haga un mal uso de estos y, al menos en España, las zonas de vuelo en ciudades y zonas con cantidades más o menos grandes de población son muy reducidas.

Estos drones poseen cámaras que pueden ser utilizadas tanto como para simple navegación como conceptos más avanzados como ‘Computer vision’ o uso de algoritmos.

Este trabajo se va a centrar en el uso de la cámara del dron para aplicar algoritmos a grabaciones que realice el UAV en un vuelo.

Sirviendo esto como una introducción a este tipo de vehículos no tripulados, en los siguientes subapartados se realizará un mayor enfoque en estos.

2.1.1 Movimiento del dron

Para que un dron pueda realizar sus movimientos de manera correcta sin tener problemas durante el vuelo, el dron usa sensores para esto y mediante la información realizar varias acciones de vuelo. En primera instancia, se explicará de manera breve la Unidad de Medición Inercial y se comentará para que sirve y los sensores que posee. Además, también se describirá los distintos tipos de movimientos que puede realizar un dron.

2.1.1.1 Unidad de Medición Inercial

Es una unidad de medición presente dentro del dron que, aunque podría ser explicado más adelante en el subapartado de sensores, será explicado aquí por tener una relación estrecha con el movimiento del propio dron y que posee una serie de sensores integrados en ella para obtener datos.

Es capaz de recoger datos a través de distintos sensores que están integrados en ella y mediante el procesamiento de esos datos con la CPU puede estimar información como la altitud, posición y velocidad.

Además, posee la capacidad de poder descartar datos que procedan de sensores que no estén funcionando de manera correcta por cualquier razón y realizar estas estimaciones sin la información procedente de ese sensor en específico.

Como bien se ha comentado, la Unidad de Medición Inercial o UMI, posee varios sensores que le permiten el cálculo de los datos comentados anteriormente. Por tanto, se va a explicar brevemente los más comunes dentro de una unidad:

- **Acelerómetro:** Mide las fuerzas gravitacionales en un sistema de coordenadas fijo.
- **Giroscopio:** Mide la velocidad angular.
- **Magnetómetro:** Mide el campo magnético local.

Mediante estos datos y su posterior procesado el dron es capaz de estimar información y utilizarla durante el vuelo para que no surjan problemas durante este. [6]

2.1.1.2 Tipos de movimientos

Antes de explicar los movimientos de un dron durante el vuelo cabe destacar los distintos perfiles de vuelo básicos que puede realizar un dron [7]:

- **Ascenso:** Es el movimiento típico de elevación, tanto para empezar el vuelo como para posicionarse en altitudes más elevadas. Para ello los motores del dron deben darles suficiente potencia a las hélices para girar a suficiente velocidad y aumentar su altitud.
- **Descenso:** Es el movimiento contrario del Ascenso y simplemente descenderá en el vuelo. Las hélices deben ir reduciendo velocidad para alcanzar la altitud deseado y mantenerse en ella.
- **Crucero:** El modo crucero se refiere a un movimiento homogéneo y rectilíneo con trayectoria horizontal. Se suele utilizar para mapear terreno.
- **Estacionario:** Consiste en mantener la altitud utilizando la misma potencia en cada hélice y mantenerse flotando sin moverse ni de manera horizontal ni de manera vertical.

Obviamente existen otras dinámicas de vuelo más complejas que combinan giros constantes, cambios de velocidad y de altura y demás pero no parten de dinámicas de vuelo en sí y podría considerarse como un perfil de vuelo libre.

Una vez comentado los perfiles de vuelo, se va a describir los distintos movimientos que realiza un dron durante su vuelo:

- **Guiñada o Yaw:** Se refiere al movimiento que puede realizar un dron en su eje vertical, pudiendo girar 360 grados de derecha a izquierda o viceversa la dirección hacia donde se desplaza. Es un movimiento muy importante puesto que permite cambiar la dirección del dron y orientar la parte frontal del dron, donde suele estar colocada la cámara, en el sentido de la dirección donde se va a realizar el movimiento.

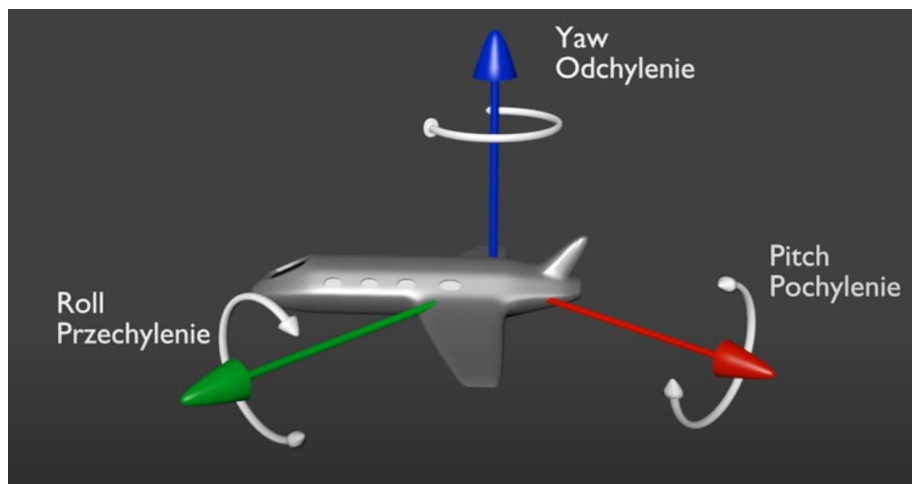


Figura 2. Movimiento de Guiñada [8]

- **Aladeo o Roll:** Este movimiento permite la inclinación del dron hacia la derecha o la izquierda y sirve para que planee realizando giros leves a derecha o izquierda.

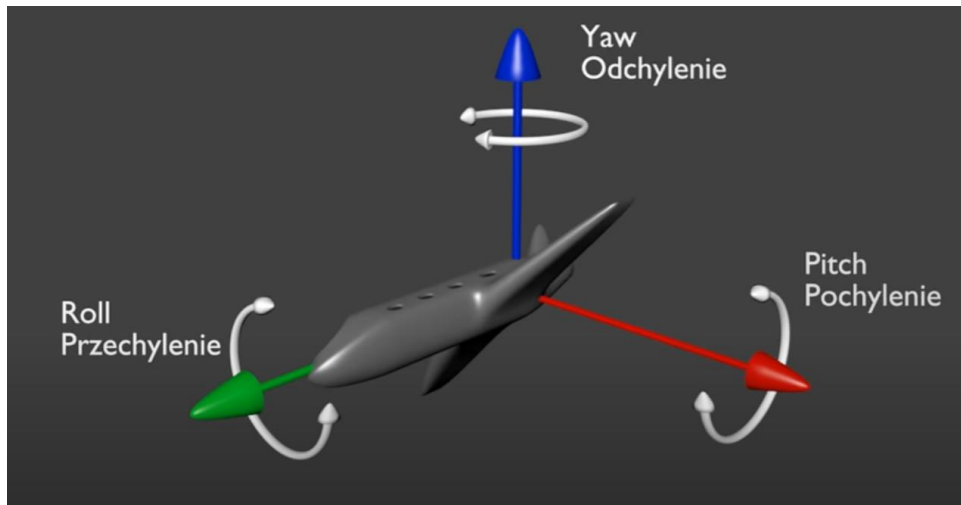


Figura 3. Movimiento de Aladeo [8]

- **Cabeceo o Pitch:** Es el movimiento análogo del aladeo, pero en vez de realizar los giros de derecha a izquierda, los realiza de arriba a abajo. Permite realizar movimientos de picado hacia arriba y hacia abajo para poder cambiar la altitud.

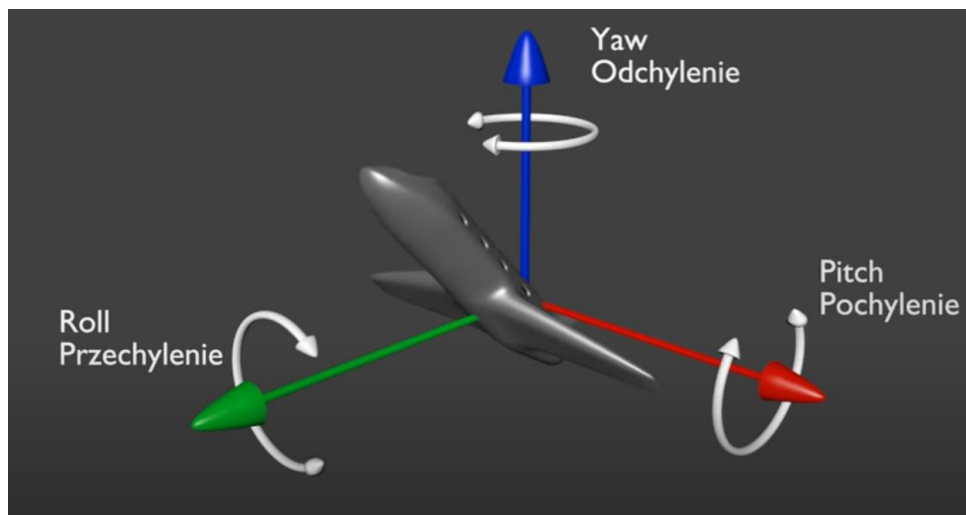


Figura 4. Movimiento de Cabeceo [8]

2.1.2 Clasificaciones de dron

Según las diferentes características podemos clasificar los drones en varios tipos según si son tripulados o no, si poseen hélices o el número de estas. Se van a explicar los distintos tipos dependiendo de estas características y su definición. [9]

2.1.2.1 Drones según si son tripulados

Podemos distinguir dos clases de definición para los drones en este tipo:

- RPA o RPAS: Son, como dicen sus siglas: Remotely Piloted Aircraft y Remotely Piloted Aircraft System, drones pilotados que necesitan de un profesional cualificado para llevar a cabo todas las tareas necesarias.
- UAV o UAS: Las siglas hacen referencia a Unmanned Aerial Vehicle y Unmanned Aerial System, por lo que son drones que no son manejados y utilizan algoritmos y programas para realizar misiones autónomas.

En este proyecto nos vamos a centrar en estos últimos ya que las misiones que se van a realizar utilizan drones no manejados por control remoto.

2.1.2.2 Drones según la sustentación

El modo de sustentación se refiere a la forma que tiene el dron para mantenerse en el aire.

Según esta forma existen dos tipos de drones:

- Drones de ala fija: Se refiere a drones que no poseen hélices y por tanto necesitan de mecanismo de ayuda para su despegue, puesto que por sí mismo es imposible realizarlo, tienen una gran autonomía de vuelo por su aerodinámica y, debido a esto, suelen ser utilizados para mapear grandes cantidades de terreno.



Figura 5. Ejemplo de dron de ala fija [9]

- Multirrotores: Si pensamos en un dron, la imagen que se nos vendrá a la mente será la de un multirrotor, son los drones con hélices en cada brazo para despegar

y controlar el vuelo. A diferencia de los drones de ala fija, este tipo de drones puede mantenerse en vuelo quietos sin tener que desplazarse. Son el tipo de dron usado en este proyecto ya que es más realista y sirven mejor para misiones autónomas del estilo que se van a realizar.

2.1.2.3 Drones según el número de brazos

Según el número de brazos que posea el multirrotor, tendrá una denominación u otra, si bien se pueden añadir más brazos y motores, los más extendidos son los siguientes:

- **Tricóptero:** Cuando poseen tres brazos, pero cabe recalcar que uno de los motores de estos sirve de servomotor y los motores delanteros giran en direcciones opuestas para ofrecer estabilidad y contrarrestarse.
- **Cuadricóptero:** Drones con cuatro motores y cuatro brazos. Son los más comunes de cara al público y son los que van a ser utilizados en este proyecto.
- **Hexacóptero:** De igual modo, este tipo posee seis motores y seis brazos. Muy útiles para fotografía, más seguros que los demás y son los más utilizados dentro del ámbito profesional.
- **Octacóptero:** Posee ocho motores y ocho brazos y es el más estable y con más potencia. También es el más pesado y por eso su uso en espacios más cerrados se complica.
- **Coaxiales:** Este tipo de dron no destaca por su número de brazos si no por el número de motores en cada brazo, posee dos motores por brazo siendo capaz de aguantar una mayor cantidad de peso.

2.1.3 Enjambre de drones

Según la cantidad de drones que se utilicen por misión si se utilizan en conjunto estos son denominados enjambre. Aunque se va a usar un único dron para realizar la misión cabe recalcar y explicar este tipo de misiones.

Para que varios drones puedan colaborar y realizar tareas de manera conjunta, estos deben hacer uso de los sensores que poseen, que serán explicados en el apartado siguiente, y poseer una programación suficientemente hábil para evitar choques entre ellos y colapsos de tareas. Si bien actualmente no existen tareas extremadamente complicadas que estos enjambres puedan realizar, ya que se encuentra todavía en desarrollo, se han conseguido vuelos armónicos para tareas como toma de fotografías o reparto de semillas para el cultivo.

Para que esto siga su curso de desarrollo los drones deberían mantener comunicaciones entre ellos durante todo el vuelo y ser conscientes de su posición y de los demás, además, de obviamente, su propia tarea a realizar. [10]



Figura 6. Enjambre de drones [10]

2.1.4 Sensores

Para que los drones puedan realizar un número de tareas mayor que simplemente vuelo o toma de fotografías estos disponen de una serie de sensores que pueden ofrecer distintos tipos de datos.

Pero antes de ponernos con los sensores en sí, cabe recalcar que los drones poseen cámaras, dependiendo del coste del mismo esta será de mayor o menor calidad, y es la herramienta con la que se suele hacer toma de fotografías, además de ser la que es usada dentro del proyecto.

Una vez comentado brevemente la existencia de la cámara y, a pesar de que no son usadas en el trabajo, es bueno al menos describir de manera escueta las posibilidades que tiene un dron común con los distintos sensores y radares que estos llevan instalados. Si bien existen un mayor número de sensores posibles dentro de un dron se van a explicar los más comunes y los que más posibilidades de uso tienen. [11]

2.1.4.1 GPS

Un sensor GPS es un sensor capaz de determinar la posición del dispositivo que lo usa. Los drones usan la posición que obtiene este sensor para combinarla con los datos obtenidos mediante la UMI (Unidad de Medición Inercial), ya comentado en el apartado de movimiento del dron, hace que se pueda controlar la trayectoria del dron. Mediante este sensor pueden ser evitadas numerosos casos peligrosos como accidentes o choques, además de poder proporcionar información como altitud del vehículo o su ubicación en tiempo real.

Otro tipo de funcionalidad que puede ser realizada mediante este sensor sería la de “regreso a casa” que establece un punto de origen y si el dron está agotando su batería o se aleja lo suficiente del piloto pueda volver a ese punto de manera autónoma.

También se puede navegar mediante puntos de referencia de coordenadas o ir obteniendo información de este estilo de manera constante, especialmente útil en situaciones de rescate, por ejemplo, al poder buscar a una persona en una zona muy amplia y poder mostrar las coordenadas en latitud y longitud de donde se encuentra. [12]

2.1.4.2 LiDAR

Toma el nombre de sus siglas en inglés Light Detection And Ranging, esta tecnología hace uso de pulsos láser rápidos para medir el tiempo que tarda entre liberar el pulso y cuando se refleja en una superficie u objeto y se recibe de nuevo el pulso, mediante este tiempo se es capaz de calcular la distancia al objeto requerido.

Al ser combinado con softwares más potentes, es posible realizar un modelado 3D del terreno y de su elevación, crear imágenes de las propias superficies y es capaz de recopilar datos mediante su uso.

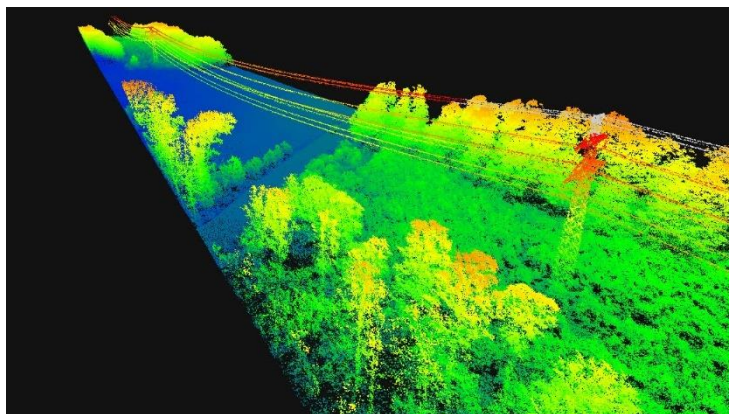


Figura 7. Ejemplo uso LiDAR [13]

2.1.4.3 Sensor térmico

Como su propio nombre indica, son sensores capaces de medir la temperatura de superficies y objetos recogiendo la firma de calor que este emite. Se recopilan estas firmas térmicas recopilan los datos para mostrarlos en modelos de color para ser reconocidos fácilmente por el ser humano usando el rojo intenso para las firmas de color de mayor temperatura y colores azulados para temperaturas más frías.

Es un sensor muy útil a la hora del uso de drones puesto que en situaciones de emergencia como rescate de personas en catástrofes o ayuda en incendios pueden llegar a salvar vidas al poder reconocer información de manera más sencilla.

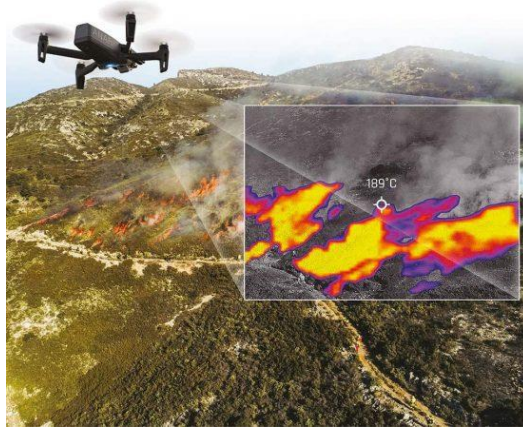


Figura 8. Ejemplo uso sensor térmico. [14]

2.1.4.4 Sensor Multiespectral e Hiperespectrales

Los sensores multiespectrales son un tipo de sensor capaz de reconocer no solo la longitud de onda de los tres colores básicos que percibe el ser humano (rojo, verde y azul) sino que también es capaz de reconocer longitudes de onda que el humano no puede percibir como la radiación infrarroja cercana y la radiación infrarroja de onda corta.

Son especialmente utilizadas en la agricultura al poder observar la salud de los cultivos en un área bastante extensa, además de poder ser utilizado en otros campos como comprobar el funcionamiento de placas solares o puntos que pueden causar problemas en un gaseoducto.



Figura 9. Comparación cámara RGB con cámara multiespectral NDVI

Por otro lado, los sensores hiperespectrales, si bien siguen la lógica de los sensores multiespectrales, son básicamente una versión bastante mejorada de estos. Son utilizados cuando la precisión de los multiespectrales se queda corta en ámbitos como estudios químicos del terreno o de minerales y son capaces de proporcionar entre cien y doscientas bandas espectrales de datos por píxel.

Otro ejemplo para poder conocer la diferencia de precisión entre estos dos tipos de sensores sería su uso dentro de la silvicultura, que es la disciplina que gestiona los bosques y las zonas forestales, las bandas que percibe un sensor multiespectral sería capaz de distinguir árboles de arbustos mientras que el sensor hiperespectral podría identificar las especies exactas de estos árboles.

2.1.5 Formas de uso

Como ya se ha comentado en varias ocasiones, los drones pueden o bien ser utilizados y controlados mediante control remoto siendo tripulados por un piloto o pueden ser autónomos que realicen misiones sin necesidad de ningún control.

Para que un dron autónomo realice todas las tareas que se le pide de manera correcta y sin complicaciones debe existir un buen software que le respalde, ya no solo en situaciones normales como vuelos o tomas de decisiones, sino también en caso de fallo de motor o situaciones de emergencia, en el que el dron debe poseer mecanismos de control y de vuelta a tierra de manera controlada para que la figura del piloto no sea necesaria. Por lo que al necesitar un software potente y bien desarrollado es más costoso su desarrollo que simplemente utilizar un piloto que se haya formado.

Por otro lado, aunque controlar los drones de manera remota pueda ofrecer quizá una mayor libertad al usuario, si se quisiera realizar operaciones más complicadas con mayor número de drones es más beneficioso utilizar drones autónomos y claro está ofrece una mayor velocidad en la realización de la tarea en sí.

Por lo tanto, los drones autónomos ofrecen más velocidad de cómputo y abaratan y facilitan tareas complicadas con estos, pero para tareas más simples o para uso personal es mucho mejor su uso mediante control remoto.

2.2 Controlador de vuelo

Un controlador de vuelo es una tarjeta integrada dentro del dron que contiene un procesador que es capaz de tomar las órdenes que le llegan al dron mediante el receptor y procesarlas para realizar la acción de manera correcta, por ejemplo, si se le pide al dron moverse hacia delante, el controlador de vuelo recibirá la orden y calculará la potencia que debe enviar a los motores del dron, la inclinación que debe tomar, el grado del giro, etc. También poseen puertos micro-USB para poder conectarlos a un computador y

configurarlo mediante programas como BetaFlight. Es decir, en general, es básicamente el componente que realiza todas las acciones del dron y que conecta los demás componentes del mismo para que el dron funcione correctamente y de una manera centralizada. [15]

Si estuviésemos en el caso de que fuese un dron real se necesitaría un controlador Pixhawk o Ardupilot para poner un ejemplo y cómo podemos ver en la Figura 10, pero como nos encontramos en un entorno simulado y la parte hardware ya se encuentra dentro de la librería AirSim este controlador de vuelo no es necesario, pero es necesaria la inclusión de un software como PX4 para que el dron funcione correctamente y de una manera realista dentro de este entorno. [16, p. 4]



Figura 10. Controlador de vuelo Pixhawk [17]

2.2.1 PX4

Es el software que ha sido utilizado en el proyecto como controlador de vuelo para la simulación del dron y es código abierto por lo que cualquiera puede utilizarlo tanto para proyectos personales como proyectos de investigación. Resulta muy útil por la gran cantidad de características que presenta:

- Arquitectura modular: Esto significa que está formada por módulos más pequeños que juntos crean el software en conjunto. Utiliza esta estructura tanto en hardware como en software y mediante la arquitectura basada en puertos, lo que significa que los usuarios pueden añadir componentes sin que se pierda rendimiento.
- Alto grado de configurabilidad: Debido a la gran cantidad de APIs y SDKs optimizados que presenta PX4 se puede cambiar un módulo por otro sin modificar el núcleo del software, haciendo que las características de este sean fácilmente reconfigurables y añadidas.
- Validado en el mundo real: PX4 ha sido probado en vuelos tanto de prueba que se realizan periódicamente como vuelos reales, probando así que también funciona de una manera correcta en el mundo real.
- Interoperabilidad: Ofrece una gran cantidad de dispositivos compatibles con el software.

- Licencia permisiva: Permite comercializar un producto que ha sido realizado mediante PX4 por lo que es muy útil también para empresas.
- Características de seguridad: Posee características como un paracaídas, modos de vuelta o un comportamiento seguro en caso de error, lo que permite utilizar drones con menos peligro.

2.2.2 Diferencias entre Ardupilot y PX4

La principal diferencia entre estos dos softwares de controladores de vuelo es el tipo de licencia que tienen, mientras que Ardupilot posee licencia GPL3, lo que significa que todos los programas o softwares derivados de Ardupilot también serán publicadas bajo esta licencia y por tanto si quieres comercializar un software, primero tienes que publicar el código fuente de manera completamente gratuita, mientras que PX4 usa una licencia BSD3 lo que significa que, si bien es código abierto y cualquiera puede usar PX4 para hacer simulaciones o cualquier tipo de pruebas con drones, los programas derivados de este no tienen por qué publicar su código fuente y por tanto es más apetecible para empresas.

Hace unos años se podía considerar que Ardupilot tenía diferencias muy grandes en cuanto a comunidad que usa el programa y por tanto existirá menos ayuda, pero actualmente ha habido un aumento en el uso de PX4 y por tanto este problema estaría solventado.

PX4 soporta un mayor número de periféricos, sensores y demás que Ardupilot, tiene una simulación software-in-the-loop más desarrollado y madura que Ardupilot y tiene una arquitectura más ubicua.

Cabe señalar que en cuanto a rendimiento ambos son más o menos parecidos y la elección de un software u otro no debería ser sobre esto porque la diferencia es mínima y por tanto la elección será más de facilidad de uso o comodidad, además de las funcionalidades que presentan. [18]

2.2.3 Conclusión de uso de PX4

Para concluir la explicación del controlador de vuelo y del porqué se ha usado este software en particular, se ha usado PX4 por lo comentado anteriormente en la comparación con Ardupilot y por su facilidad de uso dentro de Airsim y para el manejo del dron y sus tareas, se ha concluido que PX4 era el software adecuado para este proyecto.

2.3 MAVLink

Tanto en un entorno simulado como en un entorno real, los sistemas de los vehículos aéreos no tripulados necesitan señales para comunicarse y realizar las distintas tareas que pueden encomendársele.

MAVLink [19] es un protocolo de mensajería muy liviano para la comunicación con drones y entre sus componentes que sigue un patrón de diseño moderno que mezcla el patrón de publicar y suscribir y el patrón de punto a punto. El flujo de datos es enviado y publicado como un ‘topic’ mientras que los subprotocolos son enviados de punto a punto.

Presenta las siguientes características clave:

- Muy eficiente: La primera versión necesita solo 8 bytes como mucho por paquete mientras que la segunda solo necesita 14, siendo esta mucho más segura que su predecesora.
- Muy confiable: Ha sido usado desde 2009 en numerosos canales de comunicación, vehículos y estaciones proporcionando un servicio eficiente. Además, posee métodos de detección caídas de paquetes, corrupción de los mismos y autenticación.
- Gran cantidad de lenguajes de programación: Como podemos ver en la Figura 11, MAVLink da soporte a una gran cantidad de lenguajes de programación y de poder ser ejecutado en numerosos sistemas operativos o microcontroladores, además existen proyectos independientes que dan soporte a algunos extra pero estos no son parte estricta de MAVLink,.
- Numerosos sistemas de forma concurrente: Permite hasta 255 sistemas en su red de forma concurrente.
- Como ya se ha comentado, permite comunicaciones entre los propios componentes del dron y comunicaciones externas.

Language	Generator	MAVLink v1	MAVLink 2	Signing
C	mavgen	✓	✓	✓
C++11	mavgen	✓	✓	✓
Python (2.7+, 3.3+)	mavgen	✓	✓	✓
C#	mavgen	✓	✓	
Objective C	mavgen	✓		
Java	mavgen	✓	✓	
JavaScript (Stable)	mavgen	✓	✓	X
JavaScript (NextGen)	mavgen	✓	✓	✓
TypeScript/JavaScript	mavgen	✓	✓	X
Lua	mavgen	✓	✓	X
WLua (Wireshark Lua bindings)	mavgen	✓	✓	NA
Swift	mavgen	✓		
Rust	rust-mavlink	✓	✓	

Figura 11. Lenguajes de programación para MAVLink [19]

2.4 MavSDK

Es una colección de librerías para varios lenguajes de programación que sirve para interactuar con sistemas que usan MAVLink. [20]

Presenta funcionalidades como la capacidad de mover el dron o elevarlo, utilizar la telemetría de este y controlar información que maneja. Por lo que es extremadamente útil su uso en contextos de este estilo pudiendo manejar el dron para realizar misiones de vuelo y demás. También, como añadido, puede ser usado para manejar otros vehículos como coches o incluso estaciones de tierra y dispositivos móviles que capten información.

2.5 Simulación de drones

La simulación de drones es un ámbito muy útil a la hora de poder probar cualquier avance tecnológico que se quiera realizar en este campo o pruebas más simplemente orientadas a pequeños proyectos. A la hora de conseguir simulaciones realistas que se aproximen a la realidad son especialmente útiles para la creación de software que, una vez probado en estas simulaciones, puedan ser llevados al mundo real y comprobar su funcionamiento.

Se va a explicar las herramientas y programas utilizados para realizar la simulación del dron y las características de estos para conseguir que esta simulación sea realista. Además, se hará una comparación de varios softwares para la simulación de drones y se explicará por qué se ha usado AirSim en vez de otros simuladores.

2.5.1 AirSim

AirSim [21] es un simulador para drones, coches y más vehículos construido en Unreal Engine (aunque existe una versión experimental para Unity). Es muy útil para proyectos como el que se está presentando puesto que es software open-source, multiplataforma y que posee capacidad para realizar simulaciones software-in-the-loop con controladores de vuelo como PX4 y ArduPilot y simulaciones hardware-in-the-loop con PX4. Debido a esto se consiguen simulaciones física y visualmente realistas.

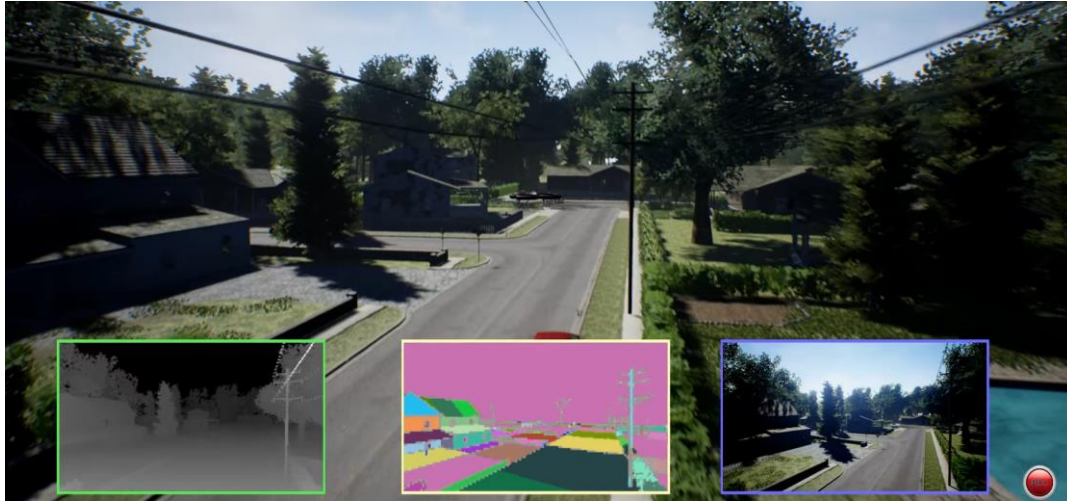


Figura 12. Ejemplo AirSim con Unreal [22]

El objetivo de este software es el de ser desarrollado y que su evolución sirva para ayudar en la investigación en el campo de la inteligencia artificial ya sea con ‘deep learning’, ‘Computer vision’ y algoritmos de aprendizaje para vehículos autónomos. Al ser su objetivo este es normal que esté disponible en varios de los lenguajes de programación más utilizados como C++, C#, Java o Python.

Existe un modo llamado ‘Computer Vision’ usado en situaciones en los que solo necesitas la cámara y no necesitas ningún tipo de dinámicas de movimiento del vehículo ni orientación de ningún tipo. En el siguiente subapartado, se explicará más a fondo el controlador de vuelo PX4, pero para hacer un pequeño avance, si bien como se ha comentado puede utilizarse sin el controlador de vuelo, pero para un correcto funcionamiento de las funcionalidades del dron es necesario las propiedades del controlador de vuelo para que reciba informaciones de la orientación a través del giroscopio y del acelerómetro, girando en torno a tres ejes y actualizando los estados de estos a los deseados. [23, p. 4]

Como detalle hay que comentar, que, aunque en el momento de realización de este proyecto AirSim era un software completamente de código abierto, en el momento de escritura de este documento Microsoft ha declarado que va a dejar de dar soporte a la opción libre para poder comercializar AirSim como producto de pago bajo el nombre de Project Airsim añadiendo varias funcionalidades como la conexión con torres 5G y su inspección. [24]

2.5.2 Comparación Airsim y otros simuladores

En el proyecto que se está describiendo el simulador utilizado ha sido AirSim, pero existen otras alternativas de simuladores y, por tanto, se va a hacer una comparación con AirSim de algunos de los más usados para determinar por qué se ha usado este.

Gazebo es otra opción dentro de los simuladores para poder realizar pruebas en un entorno simulado para realizar tareas ya comentadas.

La principal diferencia que podemos observar a simple vista entre la Figura 12 y la Figura 13 es el realismo que presentan ambas simulaciones siendo AirSim, al estar integrado en Unreal Engine, que es un motor gráfico que permite un mayor grado de realismo, hace que por esta parte AirSim sea mucha mejor opción, ya que uno de las motivaciones ya comentadas del proyecto era la de mejorar el entorno simulado para que se asemejase lo máximo posible al mundo real y por tanto AirSim es una mejor opción.

Aunque, como ya se ha comentado anteriormente, para trabajos posteriores a este si nos ponemos desde un enfoque económico o de ayuda de la comunidad, al dejar de ser código de abierto, quizás Gazebo sea una opción más apetecible desde ese enfoque.

En cuanto a las funcionalidades o realismo dentro de las dinámicas y físicas del dron, ambos son bastante parecidos teniendo más o menos la misma cantidad de sensores y dinámicas de vuelo. [25]

Aunque se esté comentando únicamente estos dos simuladores existen un sinnúmero de estos, pero se ha centrado en estos dos puesto que son los más usados y por tanto los que pueden ser más fáciles de usar y desarrollar al tener gente que pueda resolver las dudas que puedan surgir y que, además, son sistemas cómodos de usar.

Por lo tanto, la conclusión es que, si bien en cuanto a funcionalidades son más o menos parecidos, en cuanto a comunidad para resolver dudas Gazebo gana por ser de código de abierto desde sus inicios y tener una comunidad mayor, pero queda lastrado por su escaso realismo en comparación con los escenarios que son posibles dentro de AirSim y por eso en este proyecto se ha optado por esta opción.

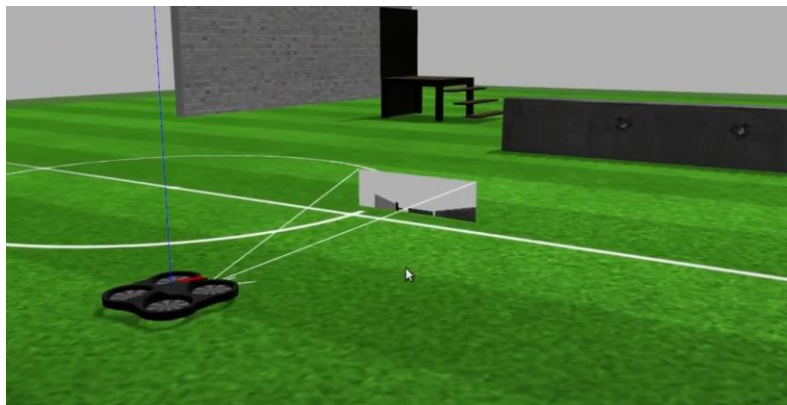


Figura 13. Ejemplo de simulación en Gazebo [26]

2.5.3 Unreal Engine

Es un motor gráfico y de juego creado por Epic Games que ha tomado mucha relevancia, especialmente en los años recientes, debido a la mejora que traído respecto a los gráficos y al realismo que aportan y debido a esto es usado no solo para videojuegos si no que se

usa en muchos aspectos como cine y televisión, arquitectura o simulaciones que es lo que se centra este trabajo.

Se utiliza este medio puesto que AirSim usa Unreal Engine para realizar sus simulaciones, disponiendo de un modo para utilizar el software ‘AirSimGameMode’ y colocando el dron en unas coordenadas que coincidan con las presentadas en el archivo settings.json este aparecerá en pantalla realizando las acciones que sean dictadas mediante código.

Lo bueno de Unreal es que debido a su alto nivel de realismo simulaciones de este tipo son más fáciles que en otro tipo de motores, además que actualmente existe una versión gratuita del mismo facilitando así su uso para proyectos más pequeños o académicos. También contiene un alto grado de portabilidad, siendo capaz los programas creados a partir de este de ser usados en múltiples plataformas como Windows, Linux, PlayStation, etc. [27]

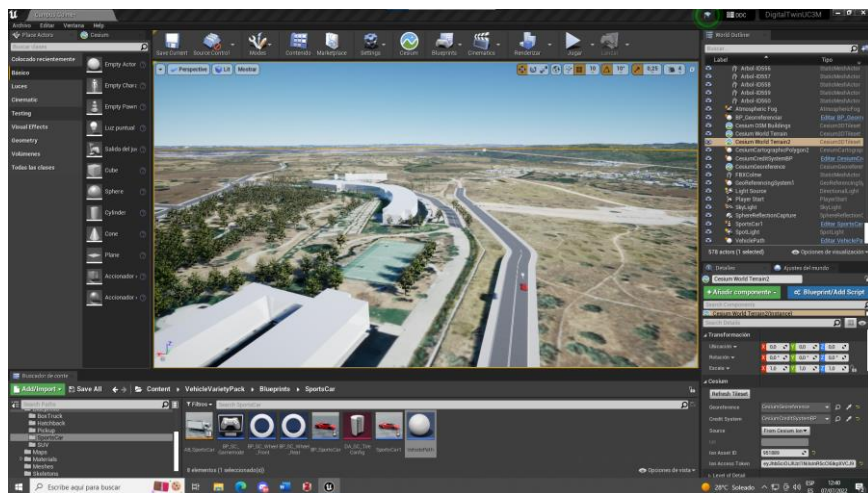


Figura 14. Ejemplo de interfaz de Unreal Engine

2.5.4 Cesium

Como se ha comentado en el apartado anterior, Unreal Engine posee numerosas opciones para hacer de las simulaciones creadas aquí de manera más realista. Una de las opciones es Cesium, una plataforma software para realizar escenarios realistas en Unreal.

Se pueden realizar copias exactas de ciudades del mundo entero si se tuviese una capacidad de computación enorme para renderizar todo. Es preciso, con gran rendimiento a la hora de usar grandes cantidades de objetos 3D como edificios, carreteras, etc. Y es de código abierto lo que facilita su uso al público. Como utiliza imágenes de satélite para coger escenarios del mundo real, los objetos, edificios y demás parecen estar aplastados si te acercas demasiado por eso estos deben ser implementados aparte. [28]



Figura 15. Ejemplo de un escenario de Cesium [29]

2.6 Lenguaje de programación

Aunque han sido utilizados otros lenguajes como C++, estos no representan el grueso del trabajo y han sido tan breves dentro del mismo que no serán explicados.

Por lo tanto, va a explicarse el lenguaje de programación utilizado en la mayoría del proyecto, una herramienta utilizada en el mismo para la visión por computador y se hará un breve paso por las ‘blueprints’ de Unreal Engine para que cuando se expliquen en profundidad las que se han desarrollado no pueda haber perdida de a que se refiere.

2.6.1 Python

Es un lenguaje de programación que nació a finales de los años ochenta y que se ha convertido en nuestros días uno de los lenguajes más utilizados. Esto se debe a varias razones:

- Licencia gratuita para que cualquier usuario pueda tanto usarlo como sacar productos sin coste al mercado.
- Un lenguaje con una curva de aprendizaje menos exponencial que otros como C o Java debido a una sintaxis más entendible y un funcionamiento más simple en algunos temas.
- Es posible utilizarlo en multitud de sistemas operativos.
- El hecho de que la licencia sea gratuita ha llevado a la creación de una cantidad enorme de librerías de todo tipo, pero Python brilla, sobre todo, en la Inteligencia Artificial con librerías OpenCV o TensorFlow que hacen que ramas como ‘Computer Vision’ o ‘Machine Learning’ sean más prolíficas en este lenguaje.

Por todo esto, se ha elegido este lenguaje de programación para este proyecto, además todas las librerías y herramientas comentadas anteriormente tienen compatibilidad con Python.

2.6.2 OpenCV

Como se ha comentado en el apartado anterior, OpenCV es una librería existente para Python, aunque está escrita en C y C++, que sirve específicamente para la rama de la inteligencia artificial de ‘Computer Vision’ y es especialmente útil para este proyecto.

Fue creada por Intel en 1999 y en la actualidad se ha convertido en una librería referente en este tipo de sistemas y sigue evolucionando hoy en día por tener un gran número de usuarios que la utiliza por ser código abierto.

Destaca en reconocimiento de objetos, análisis del movimiento de estos, capacidad para obtener información a través de vídeos o imágenes, estimación de la pose humana en un vídeo, calibración de cámaras e incluso capacidad para análisis médicos a partir de una foto de una radiografía. Como se puede observar, es una librería con un alcance enorme y que es útil en cualquier campo relacionado con imágenes o vídeos y su análisis. [30]



Figura 16. Ejemplo de uso de OpenCV con reconocimiento de rasgos faciales [31]

2.6.3 Blueprints de Unreal Engine

Las ‘blueprints’ o la tecnología ‘Blueprint’ es una interfaz basada en nodos utilizada en el motor gráfico Unreal Engine para programar elementos de un videojuego de manera que se usan para definir objetos o clases.

Se programa mediante nodos utilizando estos nodos como si fueran funciones en un lenguaje de programación siendo capaz de realizar numerosas operaciones por defecto y pudiendo realizar nuevos nodos en caso de que no existan dentro de la tecnología, bien mediante el uso de funciones combinando varios nodos o macros, que son conjuntos de nodos con parámetros de entrada y salida, o bien mediante la creación de ‘blueprints’ completamente personalizadas mediante su programación mediante C++ y usando las distintas herramientas que el propio Unreal Engine ofrece. Por ejemplo, en la Figura 17, el nodo Esperar, sería un típico wait o sleep en un lenguaje de programación clásico.

Hace falta destacar que todos estos nodos son programados a bajo nivel en C++ y por eso se pueden aumentar realizando nuevos en este lenguaje.

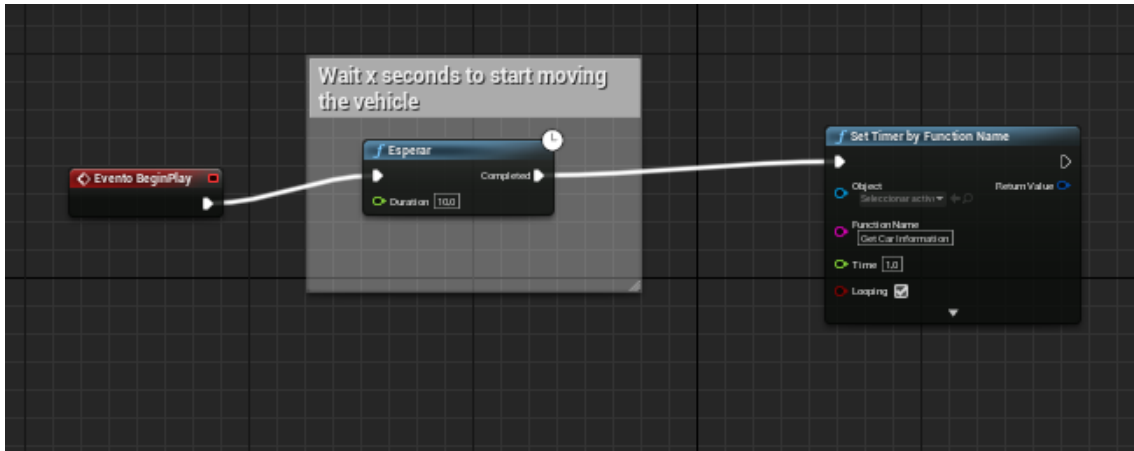


Figura 17. Ejemplo de blueprint

2.7 Inteligencia Artificial

Es una rama de la informática enfocada completamente al desarrollo de algoritmos y a la investigación de estos. Es decir, se puede encontrar en sistemas que imitan la inteligencia humana para realizar un gran número de tareas y que en cada iteración del sistema mediante información que recopilan pueden ir aprendiendo este tipo de software y mejorar por ellos mismos, lo que se llama 'Machine Learning'. [32]

Combina elementos matemáticos como la estadística y el cálculo de probabilidades para por ejemplo coger el mejor elemento en cada iteración para que el algoritmo vaya evolucionando de la mejor forma posible.

Tiene numerosas ramas dentro de la propia Inteligencia Artificial que sirven para distintas tareas, pero cabe recalcar la que ha sido utilizada en este sistema: Computer Vision

2.8 Computer Vision

Como ya se ha comentado en el apartado anterior, es una rama de la inteligencia artificial, que permite que sistemas y ordenadores puedan utilizar y procesar información a través de imágenes, vídeos y todo tipo de entradas visuales.[33]

Utiliza dos tipos de tecnologías esenciales para conseguir los objetivos que plantea este campo:

- Deep Learning: Es un tipo de Machine Learning que se basa en algoritmos donde las redes neuronales que usan aprenden de una cantidad de datos enorme.

Aprenden realizando las mismas tareas de forma repetitiva lo que lleva a una mejora progresiva en cada iteración que se realice. [34]

- Redes de neuronas convolucionales: Son una serie de redes de neuronas que fueron creadas con el fin de imitar cómo funciona el cerebro humano y está dividido en varias capas: la primera distingue formas simples, colores o bordes, la segunda capa es capaz de reconocer combinaciones de bordes y colores y la última se utiliza para clasificarlo finalmente. [35]

A través de estas dos tecnologías se puede realizar distintas tareas mediante este tipo de inteligencia artificial. En este trabajo, es especialmente útil la detección de vehículos, el trackeo del mismo a través de cada frame en un vídeo que serán explicados a continuación. Si bien los algoritmos que se usan no van a ser explicados, serán explicados en el apartado de desarrollo del proyecto, serán explicados los conceptos en sí para tener los conocimientos necesarios para que cuando más adelante se expliquen los algoritmos no haya ninguna clase de pérdida.

2.8.1 Detección de vehículos

Consiste en la utilización de algoritmos para poder reconocer objetos de todo tipo, en este caso vehículos, normalmente se utilizan máscaras, que son mecanismos para indicar que píxeles se quieren mantener al aplicarla y cuales no para poder hacer una mejor distinción en cuanto a los objetos de interés, que se aplican sobre cada uno de los frames de un vídeo.

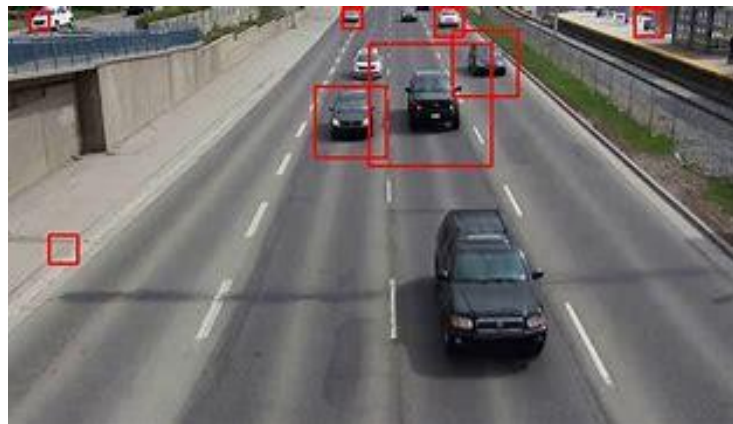


Figura 18. Ejemplo de detección de vehículos [36]

Normalmente se utilizan funciones para aplicar un ‘threshold’, que funciona para binarizar una imagen, es decir utilizar una máscara, este concepto significa que los píxeles de una imagen tomarán o un valor u otro dependiendo de la función del threshold que se use para obtener de esta manera una manera más fácil de reconocer objetos, además de la

posibilidad de eliminar ruido en un frame, que es información que queda resaltada y no es útil para el modelo de detección en sí.

2.8.2 Tracking de un objeto

Al igual que la detección de ciertos objetos es necesaria para este proyecto, como nos encontramos en un entorno en el que van a ser analizados y procesados vídeos, el tracking es una herramienta que hay que llevar a cabo. Y, esta es, la aplicación de algoritmos que sigan a un objeto a través de los frames del vídeo y que este objeto que ha sido detectado no sea considerado un nuevo objeto en cada nuevo frame. Para ello, normalmente se utilizan identificadores que se asignan a cada objeto y, si está bien realizado el algoritmo, continuará el seguimiento del objeto a lo largo del vídeo ‘trackeandolo’ mediante el centroide de la bounding box del objeto, que es básicamente un rectángulo que denota el objeto.

A través de este mecanismo pueden recopilarse datos de este objeto en el vídeo como el tiempo que permanece en pantalla o las coordenadas en esta en cada uno de los frames.

2.8.3 Estimación de la velocidad

Como se ha comentado en el apartado anterior, a través del tracking se pueden obtener datos e información que puede resultar útil para cálculos de otro tipo de información, en el caso de nuestro proyecto la velocidad de un coche al pasar por un punto del vídeo.

Uno de los problemas que surgen es la necesidad que se tiene de tener información de antemano para que el cálculo de la velocidad no resulte muy impreciso, ya que, conociendo, por ejemplo, el tamaño de la carretera mostrada en el vídeo o la distancia desde el principio de la carretera que saca el vídeo hasta el punto de interés definido anteriormente. Por lo que, si no se tiene esta información de antemano puede dar lugar a problemas de fiabilidad del sistema.

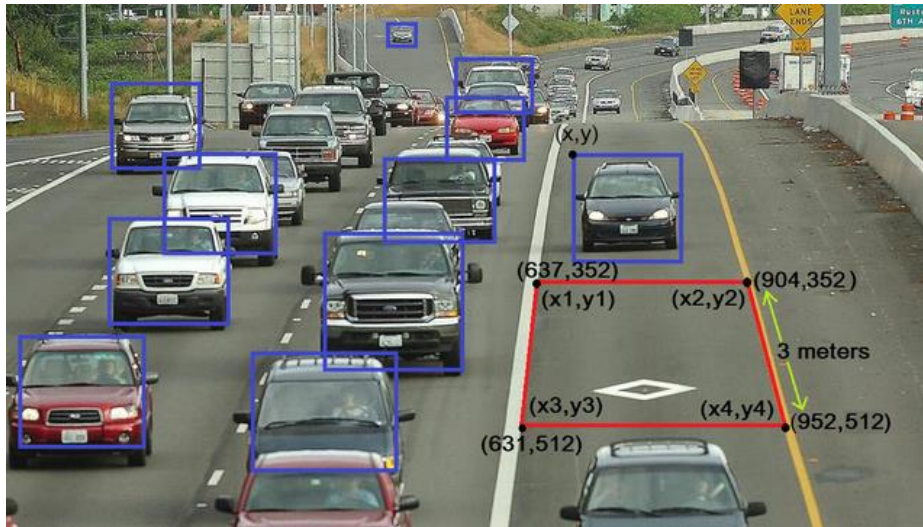


Figura 19. Ejemplo de zona donde calcular velocidad en un vídeo [37]

Otra solución que se puede alcanzar es la de tener datos geográficos de donde se ha grabado el mismo, para una precisión mucho mayor, ya que, con datos como las coordenadas geográficas de la carretera, datos como la velocidad en un tramo de la misma podría ser calculada, aunque de todos modos al tener estos datos también debería ser calculado el campo de visión de la cámara y estimar cuando entra el coche en este campo de visión para así utilizar estos datos geográficos de manera más exacta, puesto que el uso de datos geográfico sin el campo de visión exacto acabaría siendo igual que el método anterior y por eso esto se complica.

Por estas razones resulta más complicado de lo que podría parecer a simple vista el cálculo de esta información sin tener ningún tipo de datos de antemano o mediante el uso de algoritmos que calculan el número de píxeles por segundo, lo cual es algo ineficiente si no tenemos información de cuánto equivale un píxel en metros y para ello deberíamos saber altura de la cámara respecto al suelo, coordenadas geográficas, la distancia de la carretera mostrada en el vídeo, el ángulo de la cámara y numerosos datos que complican sobremanera este método.

2.8.4 Conceptos importantes del sistema

Una vez hecha una breve explicación de las tareas que se manejan en el sistema en sí, aunque más adelante se explicarán como se han aplicado estas acciones dentro del sistema y ahondando un poco más en ellas, se pasará a explicar conceptos que se han usado para el algoritmo realizado en el proyecto.

2.8.4.1 Binarización de una imagen

La binarización o umbralización de una imagen consiste en establecer un umbral por el que si un píxel es mayor en cuanto a color a ese punto límite se pondrá el valor máximo establecido y si se encuentra por debajo del umbral será 0 como se puede ver en la Figura 20. Es decir, consiste básicamente en convertir todos los píxeles de una imagen en dos valores única y normalmente es usada para pasar las imágenes a blanco y negro, puesto que de este modo es más sencillo el reconocimiento de objetos.

$$\text{dst}(x, y) = \begin{cases} \text{maxval} & \text{if } \text{src}(x, y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Figura 20. Fórmula de la binarización de una imagen [38]

Existen distintos métodos de binarización estando el mostrado en la imagen anterior o la binarización inversa como vemos en la siguiente imagen:

$$\text{dst}(x, y) = \begin{cases} 0 & \text{if } \text{src}(x, y) > \text{thresh} \\ \text{maxval} & \text{otherwise} \end{cases}$$

Figura 21. Binariación inversa de una imagen [38]

Pero, también existen otros métodos como Otsu [38] que no necesita de establecer un umbral si no que determinar cuál es el umbral óptimo para la imagen. Esto lo hace mediante la asunción de que solo existen dos tipos de píxeles: los que se encuentran al frente de la imagen y los que están en el fondo de la misma. Con esto calcula el histograma de la escala de grises de la intensidad de los píxeles de la imagen, separándose en dos picos, como podemos ver en la Figura 22, y estos dos picos siendo tanto el fondo como frente de la imagen y mediante esto calcula el valor óptimo del umbral. De todos modos, si bien el método Otsu es útil y calcula el valor sin la necesidad de saber ningún dato no tiene ninguna forma de saber si los pixeles pertenecen realmente al fondo de la imagen o al frente más allá de esta separación por picos en el histograma de la escala de intensidades.

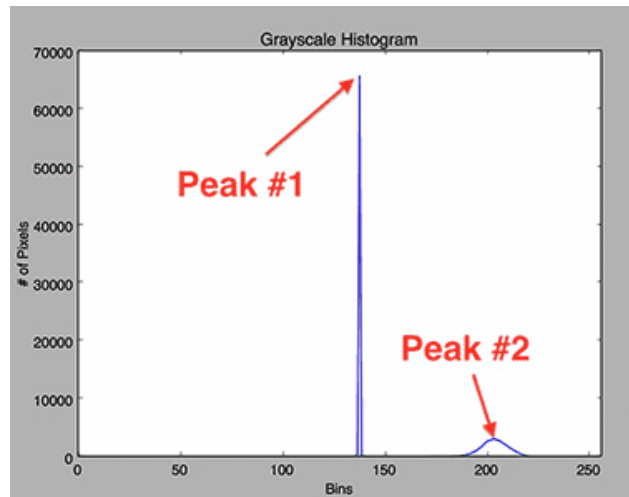


Figura 22. Histograma Otsu

Para terminar, se va a mostrar una imagen binarizada de un vehículo al lado de la imagen real para que se observe mejor este fenómeno, viendo como todos los píxeles de la imagen pasan a tener un color u otro, en nuestro caso blanco y negro ya que como se ha comentado favorece la detección de vehículos:

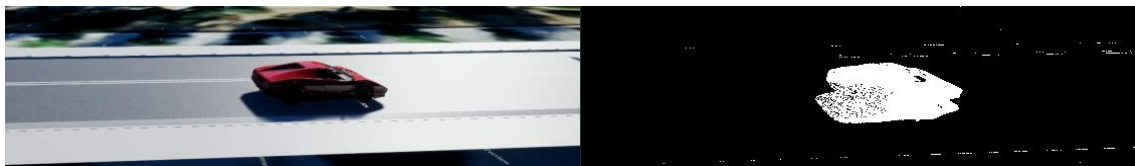


Figura 23. Comparación imagen binarizada con imagen real.

2.8.4.2 Algoritmo KNN

Es un algoritmo que toma su nombre del inglés K-Nearest neighbors que entra del tipo de algoritmos llamados de aprendizaje supervisado. ¿Esto qué significa? Pues bien, son algoritmos que reciben un conjunto de datos que están etiquetados con los valores de salida sobre los que puede entrenarse y mediante esto definir un modelo de predicción.

Una vez sabiendo esto, el funcionamiento del algoritmo es el siguiente:

- Establecer el número de K vecinos.
- Calcular la distancia posible entre ellos (Euclídea, Manhattan...)
- Tomar los K vecinos más cercanos dependiendo de la distancia calculada.
- Entre los K vecinos ir contando el número resultante de cada etiqueta o categoría de los mismos.
- Atribuir un nuevo punto a la categoría más presente entre los vecinos más cercanos.

Con esto se puede clasificar o predecir comportamientos entre conjuntos de datos, incluyendo píxeles en un entorno y por esto mismo se usa dentro de uno de los sustractores de fondo más utilizados. [39]

2.8.4.3 Sustractor de fondo

Un sustractor de fondo es uno de los enfoques más populares a la hora de realizar detección de objetos en imágenes y funciona comparando partes del vídeo o del frame que se mueven respecto al fondo o al frente del mismo.

Este método es usado para encontrar objetos en el frente del vídeo de manera que los compara con frames donde no hay objetos presentes, de esta manera encontrará las diferencias entre ellos y creará una matriz de distancia.

Comparará la diferencia de estas dos situaciones y el umbral que está establecido, si es mayor se determinará como un objeto en movimiento, como podemos ver en la Figura 24. Cabe mencionar que el umbral se predefine usando los primeros frames del vídeo. [40]

```
Step# 1:  B ← Initialize frame as background [B(x, y, t)]
Step# 2:  I ← Input frame [I (x, y, t)]
Step# 3:  If Difference (I, B) > Threshold value Then
           | Return (foreground object exists)
           Else
             Return (No foreground object exists)
```

Figura 24. Algoritmo de un sustractor de fondo

Una vez explicado esto, en el proyecto se ha utilizado un sustractor del entorno basado en el algoritmo KNN y por tanto es útil explicar los parámetros que utiliza dentro de un punto de vista teórico:

- **History:** Es el número de frames usados para realizar el modelo estadístico del entorno. Si es pequeño se tomará de manera más rápida los cambios en el entorno dentro del modelo.
- **Dist2Threshold:** Es el umbral que define si un píxel pertenece al fondo o no. Según lo grande que sea será menos sensible al movimiento en el vídeo y si es lo suficientemente grande puede que no detecte absolutamente nada ya que nunca será mayor y entonces todo pertenecerá al fondo y si es muy pequeño cualquier mínimo movimiento será considerado como un objeto, por lo que lo óptimo es encontrar un equilibrio.

- **detectShadows:** Como dice su nombre es un método para detectar las sombras de la imagen y será mostradas en la máscara como un gris como se puede observar en la siguiente imagen. [41]



Figura 25. Sombras en una máscara

2.9 Soluciones existentes

Uno de las soluciones existentes al problema planteado es la que se plantea en este documento académico, “A Vision-Based Target Detection, Tracking, and Positioning Algorithm for Unmanned Aerial Vehicle” de Xin Liu y Zhanyue Zhang [42], en el que proponen un algoritmo que vendría a solucionar la estimación de la posición de los vehículos detectados mediante la cámara de un dron.

Esta solución utiliza Yolov4 como base para luego entrenar su propio dataset para la detección de los vehículos y el algoritmo DeepSORT para el tracking de los mismos y una combinación de dos algoritmos para obtener la posición del vehículo.

A la hora de comparar esta solución con la que se ofrece en este documento hay que destacar varios aspectos que hacen que se haya tomado estas decisiones y no las de este documento.

Primero, la utilización de Yolo, antes de leer este documento, se probó varios algoritmos basados en este software y no dio resultados satisfactorios, por lo que, ya era un punto negativo para su implementación de esta manera.

Otro punto a resaltar es que utiliza información proveniente del GPS y del IMU para realizar el posicionamiento a través de la posición del dron y mediante estos sacar una estimación del campo de visión del dron y conseguir obtener la distancia del dron a cada uno de los vehículos que se encuentren presentes en el vídeo. Pero, el GPS y el IMU tienen cierto margen de error que hacen que pueda variar la información sobre la latitud, longitud y la altura del dron y por tanto no dar una información lo suficientemente verídica, esto lo solucionan mediante la combinación de dos algoritmos, IMM y PF, pero en el caso que se presenta en nuestro documento, siendo en un entorno simulado donde

la información de las coordenadas tiene incluso mayor margen de error puesto que hay que sumarle el propio Cesium, por lo que no valdría del todo.

Además, los resultados que se obtienen según el documento académico son el posicionamiento respecto al dron, es decir, los metros en X e Y respecto a la posición del dron y para el caso presentado aquí se necesitaba tener una coordenadas geográficas para poder realizar los cálculos de manera más sencilla y no tener que juntar estos dos ejes con respecto al vehículo aéreo en los coches con el ángulo de la cámara, el cual también tiene error pues presenta cierta inestabilidad al encontrarse en vuelo y conseguir la distancia recorrida en el eje XY respecto al ángulo con todos los errores que puede presentar.

Retomando lo que se ha comentado justo antes, es una solución mucho más complicada que la que se presenta en este documento, teniendo complejas operaciones matemáticas para la realización de los algoritmos pero que a la hora de pasarlo a código se complicaba sobremanera.

Otro motivo de porque no se ha utilizado, es porque una idea de este sistema es que pudiese ser utilizada en cualquier vídeo mediante computadores de gama media, siempre que se tuviese información de antemano, pero para entrenar el modelo de detección que han usado basado en Yolo han usado un ordenador bastante potente con 32 GB de RAM, una gráfica muy potente y un muy buen procesador, haciendo así que no sea accesible ni para el computador utilizado para el sistema presentado en este documento ni para otros que pudieran usar nuestro sistema.

Otra solución parecida a la nuestra es la que se presenta en “Vehicle Tracking Using Deep SORT with Low Confidence Track Filtering” de Xinyu Hou, Yi Wang y Lap-Pui Chau [43].

Esta solución utiliza de nuevo DeepSORT para detectar vehículos con confianza, es decir, vehículos que se vean en posiciones extrañas o en frames que no se vean en su totalidad, esto se hace para poder aumentar los casos de detección verdaderos cuando se tiene poca confianza en que sea un vehículo y para disminuir los falsos positivos con mucha confianza.

Si bien la idea parte desde un punto de vista interesante, utilizando DeepSORT para la detección y el tracking usando unos pesos entrenados mediante un dataset de vehículos de China [44], lo que se buscaba aquí no era la detección de vehículos con poca fiabilidad si no casos más simples de visualización de carreteras de manera completa y complicarlo tanto para mejorar los resultados del sistema de manera muy anecdótica no se creyó adecuado.

Lo que si se probó de este sistema fue el uso del dataset, que como se comentará en el apartado de alternativas no dio resultados óptimos.

También ha habido otras soluciones encontradas, pero se ha querido comentar únicamente estas dos porque eran las dos más con algoritmos más interesantes y así se podía hacer una comparación con nuestro sistema y el por qué no se han tomado las vías presentadas en estas soluciones.

3 PLANIFICACIÓN Y PRESUPUESTO

Para una buena realización del proyecto es necesario una buena planificación inicial del mismo, si bien esta planificación inicial puede no seguirse al final por motivos varios y acabar con una completamente distinta, y el seguir una buena metodología de trabajo para que este no se convierta en algo realmente caótico. En este apartado se comentarán y definirán los siguientes aspectos:

- Metodología de trabajo seguida en el proyecto.
- Planificación del mismo junto a las variaciones sufridas y sus motivos.
- Presupuesto necesario en caso de llevar a cabo el proyecto de forma real.

3.1 Metodología de trabajo

A la hora de realizar el proyecto no se ha seguido en si una metodología propiamente dicha si no que se han seguido algunas directrices y formas de realizar el proyecto, pero sin tener en cuenta una metodología al uso.

Se pueden destacar varios apartados como que métodos se han usado para reconocer las funcionalidades que se iban a desarrollar en el sistema, cómo se ha revisado el documento para que este sea lo más fidedigno en lo que se refiere a un texto científico o divulgativo o cómo se ha recopilado la información usada. Pero, se van a aunar en un solo apartado siendo comentado de forma general.

La metodología usada para recopilar información útil para el proyecto, ya sea ‘papers’ o tutoriales de software como ‘Unreal Engine’, que hacen que el proyecto sea realizable, ha sido mediante la búsqueda de estos leyéndolos por encima y de manera más general y si estos podían ser útiles para el proyecto se daría paso a una lectura o comprensión mejor del medio encontrado, ya que inicialmente se ha probado multitud de posibles soluciones o informaciones que no han acabado siendo exactamente lo que se quería.

En cuanto a la identificación de requisitos del sistema, se tardó algo más porque al principio del proyecto cabe recalcar que no se tenía muy claro cuál era el objetivo final del mismo. Además de posibles adiciones futuras que comentaba el tutor en varias reuniones que hicieron que esta tarea resultara más difícil de lo común porque podía resultar lioso en algunas situaciones. Pero, una vez asentados los objetivos de manera clara, se consiguió tener unos requisitos correctos que declarasen todas y cada una de las funcionalidades que iban a existir dentro del sistema.

Para la validación y revisión de la memoria se ha seguido una metodología típica, teniendo la primera revisión cuando el documento estaba avanzado lo suficiente para que esta fuese beneficiosa, y, al revisar, se corrigieron errores y se añadieron posibles mejoras para el documento, teniendo revisiones hasta obtener una memoria limpia y clara.

Y, para terminar, como método de asignar tareas y dejar constancia de estas siendo estas comentadas en reuniones con el tutor, se ha usado la herramienta ‘Trello’ que se basa en

la metodología ágil Kanban, que, en términos generales, significa dividir las tareas en tres marcos de acción, las tareas a realizar, las que se encuentran en progreso y las que ya han sido realizadas. En el caso de como se ha llevado a cabo en el proyecto, como se puede observar en la Figura 26, se ha dividido en tareas a realizar en tres subapartados: Memoria, Código de Python y tareas del ‘Unreal Engine’, lo demás permanece igual que en un tablero base de ‘Trello’.

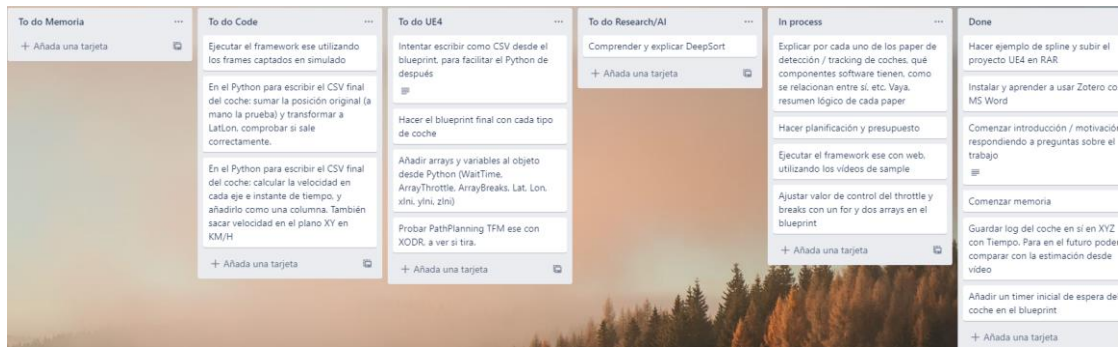


Figura 26. Uso de Trello en el proyecto

3.2 Planificación del proyecto

Como ya se ha comentado brevemente al inicio de este apartado, es importante realizar una planificación inicial para tener inicialmente un camino a seguir, pero pueden surgir ciertos problemas y motivos que hagan que o bien se tenga que retrasar o modificar esta planificación inicial para que cuadre mejor con lo que se está haciendo. Por lo tanto, se va a exponer la planificación inicial existente en este proyecto, la planificación que finalmente se ha llevado a cabo y los motivos de su modificación.

3.2.1 Explicación fases del proyecto

Aunque posteriormente van a ser señaladas dentro del diagrama de Gantt y detalladas en cuanto a número de días que se han necesitado en cada una, es necesario hacer una explicación de las fases del proyecto para que queden claras a la hora de mostrarlas en los puntos siguientes.

- **Contacto y formulación de la propuesta:** Esta fase incluye todo el tiempo que se requirió para contactar con el tutor por el tema general encontrado en el tablón de Trabajos de Fin de Grado de la universidad y la discusión llevada a cabo para determinar los objetivos propuestos.
- **Estudio inicial sobre las tecnologías usadas:** Se refiere a la fase inicial del proyecto, desde la instalación de todo lo necesario para su realización y comprobación de su correcto funcionamiento hasta pruebas pequeñas de

funciones de MavSDK como pruebas de vuelo o fotografía para ir familiarizándose con la tecnología pertinente.

- **Desarrollo:** Es la fase donde el proyecto pasa a desarrollarse realizando todas las funcionalidades propuestas.
- **Investigación:** Se refiere a todo el proceso de investigación y búsqueda tanto de conceptos para poder realizar una correcta explicación de todo en la memoria, como de algoritmos, funciones, cómo se usa la tecnología que se iba a emplear, etc. De este modo, como la investigación también conlleva la búsqueda de información para su explicación dentro de este documento, esta fase se va a entremezclar tanto con la fase de desarrollo como la fase de memoria.
- **Memoria:** Es el tiempo que se ha dedicado a la realización de este documento y todos los gráficos y explicaciones necesarias para describir de manera correcta el proyecto que se ha realizado.
- **Revisión:** Se refiere a la fase de revisión de la memoria donde se entrega al tutor una copia de la misma y este señala posibles errores o mejoras que pueden llevarse a cabo para posteriormente arreglarse.

3.2.2 Planificación inicial del proyecto

A continuación, se muestra el diagrama de Gantt de la planificación dispuesta al inicio del proyecto y que se intentó seguir para la planificación.

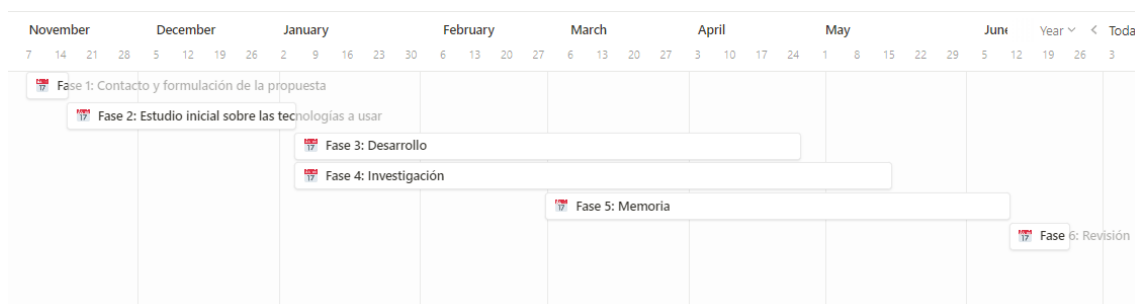


Tabla 1. Diagrama de Gantt de la planificación inicial

En caso de que por cómo está diseñada la plataforma Notion para las líneas de tiempo, que hace que algunos títulos si no ocupan el suficiente número de días no entren dentro del cuadro de cada elemento del diagrama, se muestra también la tabla con la que ha sido realizada el diagrama que indica también el número de días por tarea.

Milestone	Dates	Days
Fase 1: Contacto y formulación de la propuesta	November 6, 2021 → November 15, 2021	9
Fase 2: Estudio inicial sobre las tecnologías a usar	November 15, 2021 → January 4, 2022	50
Fase 3: Desarrollo OPEN	January 4, 2022 → April 25, 2022	111
Fase 4: Investigación	January 4, 2022 → May 15, 2022	131
Fase 5: Memoria	February 28, 2022 → June 10, 2022	102
Fase 6: Revisión	June 10, 2022 → June 23, 2022	13

Tabla 2. Planificación inicial del proyecto

3.2.3 Planificación real del proyecto

Si bien se intentó seguir una planificación fija desde el principio, fue necesario el movimiento de ciertas fases a fechas posteriores. Como muestra el siguiente diagrama de Gantt:

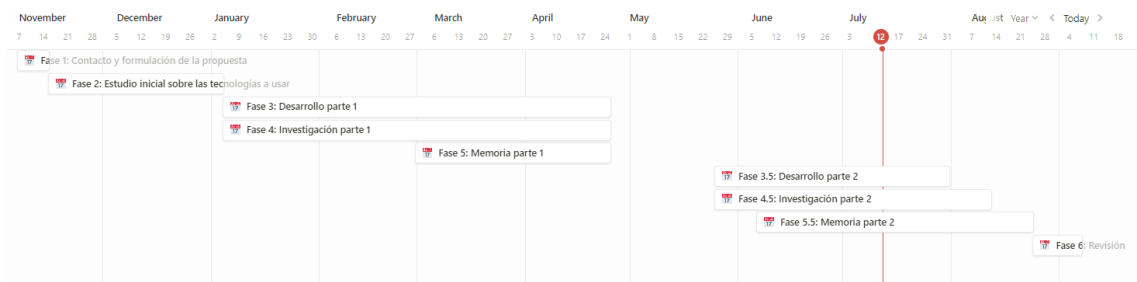


Tabla 3. Diagrama de Gantt de la planificación real

Y, de nuevo, se mostrará la tabla con la que ha sido realizado el diagrama para una mayor claridad.

Milestone	Dates	Days
Fase 1: Contacto y formulación de la propuesta	November 6, 2021 → November 15, 2021	9
Fase 2: Estudio inicial sobre las tecnologías a usar	November 15, 2021 → January 4, 2022	50
Fase 3: Desarrollo parte 1	January 4, 2022 → April 25, 2022	111
Fase 3.5: Desarrollo parte 2	May 25, 2022 → July 31, 2022	67
Fase 4: Investigación parte 1	January 4, 2022 → April 25, 2022	111
Fase 4.5: Investigación parte 2	May 25, 2022 → August 12, 2022	79
Fase 5: Memoria parte 1	February 28, 2022 → April 25, 2022	56
Fase 5.5: Memoria parte 2	June 6, 2022 → August 24, 2022	79
Fase 6: Revisión	August 24, 2022 → September 7, 2022	14

Tabla 4. Planificación real del proyecto

Es decir, el número total de días empleados en el proyecto han sido un total de **275 días**, teniendo en cuenta los días desde el inicio del proyecto, el 6 de noviembre de 2021, hasta final de la primera mitad, el 25 de abril, incluyendo los días de la segunda mitad, del 25 de mayo al 7 de septiembre, dando como suma esta cantidad de días.

3.3 Presupuesto

Como en cualquier proyecto, el presupuesto necesario para llevarlo a cabo es obligatorio que sea mostrado y explicado con todos los componentes desde físicos a componentes software para saber una estimación más o menos real de cuánto podría llegar a costar la realización de este sistema. Para ello, se van a mostrar una serie de tablas y cálculos de gastos para tener esta estimación del presupuesto de una manera más visual.

Se va a dividir en varios tipos de costes, que incluye los costes materiales, costes de trabajo humano, costes indirectos y demás.

3.3.1 Costes materiales

Empezando por los costes materiales, como la licencia de todo el software usado durante la totalidad del proyecto es gratuito solo se va a incluir los costes de hardware, los cuales se encuentran presentados en la siguiente tabla:

Componente	Modelo	Coste (euros)
Procesador	AMD Ryzen 5 2600	165,50
Memoria RAM	16 GB DDR4	148,11
Fuente de alimentación	EVGA 600 W1	54,89
Tarjeta gráfica	GeForce GTX 1050 TI	174,90
Disco duro 1	SSD 120 GBs	29,04
Disco duro 2	HDD 1 TB	45,04
Monitor 1	BenQ GW2470H	125,60
Monitor 2	AOC G2590FX	199,00
Placa Base	ASUS Prime B350-PLUS	88,88
Teclado	Teclado Mecánico AKKO	100,00

Ratón	Logitech G203	21,99
Total		1152,95

Tabla 5. Costes materiales

3.3.2 Costes personales

También se han de tener en cuenta los costes en cuanto a salarios que podría haber en caso de la realización del proyecto, como este solo ha sido realizado por un programador se estimará estos costes como los de un único empleado. Tomando como referencia un sueldo de un ingeniero según la Resolución del 10 de marzo de la Dirección General de Tráfico con respecto a las tablas salariales, que presenta un salario de 1.832,07 € [45] y tomando este como un sueldo de jornada completa se queda como 11,45 euros la hora y como se ha relatado en la planificación final del proyecto, se han trabajado un total de 275 días, a una media de 2.5 horas el día, puesto que ha habido días con más de diez horas de trabajo, pero otros con jornadas más relajadas y por tanto se considera esta una buena media de horas trabajadas por día, y entonces el coste se relata en la siguiente tabla:

Coste por hora (euros)	11,45
Horas empleadas	687,5
Coste total (euros)	7.871,88

Tabla 6. Costes personales

3.3.3 Costes totales

Para terminar, se mostrará una tabla que explique los costes generales de todo el proyecto, aunando los costes mencionados en los apartados anteriores con otro tipo de costes como los costes indirectos, el margen de beneficios y el impuesto de valor añadido, que en el caso de los programas y servicios informáticos es de un 21% [46]. Todo esto se encuentra enunciado en la siguiente tabla:

Concepto	Coste (euros)
Costes directos	9024,83
Costes indirectos (10%)	902,48
Margen de beneficios (25%)	2256,21
Impuesto del Valor Añadido (21%)	1895,22
Costes totales	14078,74

Tabla 7. Costes totales

Total: CATORCE MIL SETENTA Y OCHO EUROS CON SETENTA Y CUATRO CÉNTIMOS.

4 MARCO REGULADOR

A continuación, se va a explicar cualquier elemento legal que pueda influir tanto en la realización del proyecto y él porque se ha llevado a cabo en un entorno simulado como su posible aplicación en un entorno real. Se explicará la legislación española y la europea con respecto a los drones y todas sus restricciones ya que una de las razones de que sea en un entorno simulado es la poca capacidad de vuelo que existen en zonas urbanas y por tanto no es posible probar como debería ser posible las acciones llevadas a cabo en este proyecto. Además, para terminar, se hará una mención a Ley Orgánica de Protección de Datos y el problema en este proyecto si se hiciese en el mundo real.

4.1 Legislación española

Como ya se mencionado, el proyecto se centra en una aplicación de la solución en un entorno simulado, pero si se pudiese avanzar lo suficiente en esta solución, podría llegarse a una aplicación de esta en el mundo real, donde existen ciertas restricciones y normas para que no haya un descontrol con el uso de estos vehículos aéreos.

Actualmente en España ha dejado de estar vigente la normativa nacional desde el 1 de enero de 2022, pero con excepciones en torno a los clubes y asociaciones de aeromodelismo que realizan operaciones tanto con RPAS (Remotely Piloted Aircraft System) que se refiere a drones pilotados remotamente como a UAVs (Unmanned Aircraft Vehicle), drones no tripulados ni pilotados que usan sistemas autónomos, esta excepción estará presente hasta el 1 de enero de 2023 cuando se adoptará por completo la normativa europea.

Aunque cabe destacar un tipo de actividades que no siguen la normativa europea, las actividades no EASA [47], que se refieren a las actividades realizadas por servicios militares, de aduanas, policía, búsqueda y salvamento, lucha contra incendios, control fronterizo, vigilancia costera o similares, es decir cualquier tipo de actividad realizada para la seguridad bajo el control de un Estado europeo. Este tipo de operaciones son de especial interés para nuestro proyecto, puesto que lo más probables es que en una versión más avanzada sean usados por estos, aunque también puede ser usados por organismos de investigación que no entrarían dentro de este grupo. Este grupo no requiere dar de alta las actividades que lleven a cabo en el Registro de Operadores de AESEA, aunque si deben tener la cualificación pertinente para volar un dron y seguir todas las pautas de seguridad.

Para los demás tipos de operaciones como actualmente la normativa que se aplica en España es exactamente igual a la legislación europea en el siguiente apartado se explicará esta.

4.2 Legislación europea

Desde la inclusión de esta legislación se han dejado de distinguir los vuelos por ocio o profesionales y por trabajo, incluyendo ambos en la misma categoría. También existe una distinción entre tres categorías que serán explicadas a continuación. [48]

4.2.1 Categoría abierta

En esta categoría se engloban todos los vuelos que poseen un bajo nivel de riesgo, teniendo varias restricciones claras:

- Prohibido el sobrevuelo de grupos de personas.
- No está autorizado el transporte o arrojo de materiales o mercancías peligrosas.
- No están permitidas las operaciones autónomas.

Además de existir los siguientes requisitos para poder realizar vuelos dentro de esta categoría:

- Se debe tener un mínimo de 16 años.
- El operador del dron debe estar registrado en el Registro de Operadores de AESEA.
- Aprobar una formación teórica y un examen para certificar que se tienen los conocimientos necesarios para el vuelo de un dron.
- Mantener el dron siempre en la línea de visión.
- La altura máxima de vuelo permitida es de 120 metros.
- La masa máxima al despegue del dron será de menos de 25 kilos.

Para finalizar esta categoría cabe mencionar que existen tres subcategorías dentro de la misma dependiendo del peso A1 (menos de 250 gramos), A2 (menos de 4 kilogramos) y A3 (menos de 25 kilogramos).

4.2.2 Categoría específica

Esta categoría incluye a vuelos que no pueden ser englobados dentro de la categoría abierta por su riesgo. Las características para que un vuelo sea considerado dentro de esta categoría son las siguientes:

- Vuelos más allá de la línea de visión.
- Operaciones de más de 120 metros de altitud.
- Drones de más de 25 kilogramos.
- Vuelos urbanos con drones de más de 4 kilogramos.
- Arrojo de materiales.
- Vuelo sobre grandes grupos de personas.

Además, deben cumplirse los requisitos de edad, encontrarse registrados en el Registro de Operadores y un estudio sobre operaciones con riesgo.

4.2.3 Categoría certificada

Son los vuelos que más riesgo conllevan y tienen la siguiente serie de requisitos:

- Drones certificados bajo el Reglamento Delegado UE 2019/945.
- Cuando se vuele por encima de grupos de personas con un dron de más de 3 metros de envergadura.
- Cuando se vuele sobre aglomeraciones de personas; transporten mercancías peligrosas con alto riesgo en caso de accidente; o cuando implique el transporte de personas.
- Si el estudio de seguridad mencionado anteriormente indica la necesidad de la certificación del UAS y del operador y la obtención de una licencia de piloto.

4.3 Ley Orgánica de Protección de Datos

Otro de los problemas que surgiría a la hora de realizarlo en un entorno real sería la posible vulneración de la Ley Orgánica de Protección de Datos [49]. Ya que para la realización de este proyecto se necesitaría la grabación de carreteras, coches con sus matriculas, además de la posibilidad de que saliesen caras tanto de adultos como de menores lo que podría vulnerar esta ley al poseer estos vídeos sin autorización de la gente que sale en ellos y sin ser una autoridad del Estado Español.

4.4 Multas por Inteligencia Artificial

Y, para terminar este capítulo, habría que preguntarse si la legislación española ampara que la toma de decisiones por parte de un algoritmo de Inteligencia Artificial pueda acabar en multa.

Si bien los drones pueden detectar infracciones como conductores que no llevan el cinturón de seguridad o la utilización del teléfono móvil al volante, esta detección no es siempre fiable, puesto que puede fallar por diversos motivos como por ejemplo si el conductor llevase una prenda del color del cinturón lo que podría burlar este sistema.

De todas maneras, por muy avanzado que esté el algoritmo usado, actualmente se deben revisar manualmente todas los posibles positivos en cuanto a infracción se refiere por estos posibles errores que pueden surgir. Es decir, no puede multar de manera directa la Inteligencia Artificial, si no que primero debe pasar una revisión por parte de las autoridades competentes. [50]

A pesar de que la ley si incluye el uso de Inteligencia Artificial, suelen ser en casos para agilizar la recogida de la información pertinente para tramitar la denuncia al ciudadano. Incluso se llega a permitir que haya algoritmos para la redacción de la propuesta de resolución, pero solo en caso de que no exista una oposición por el ciudadano. Sigue sin existir un método de multas directas hacia el ciudadano, por lo que no sustituye de ninguna manera el trabajo de personal gubernamental si no que lo complementa para tener más información y rapidez a la hora de tramitar estas multas. [51]

De todos modos, en cuantos a multas de tráfico que es lo que nos referimos en este documento, cada posible positivo como se ha comentado ya, debe ser revisado por agentes de tráfico y comprobar la veracidad de este positivo y el contexto del mismo.

5 ENTORNO SOCIO-ECONÓMICO

Este proyecto tiene como fin la monitorización de carreteras mediante el análisis de la información obtenida por la cámara del dron. Esto, por tanto, tiene impacto en varios temas desde sociales, al final se pretende controlar la posible mala praxis de la población en la conducción, a económicos con posibles reducciones de presupuesto.

En este apartado se va a realizar una breve explicación del impacto que podría tener este proyecto en la sociedad comparando con usos de drones actualmente activos.

5.1 Impacto Social

Como se acaba de comentar, la finalidad del proyecto, si se llegase a llevar a un entorno real y no solo en un entorno simulado, es monitorizar la conducción de la población y controlarla para que se produzcan menos infracciones y accidentes. Por lo tanto, teniendo esto en cuenta, se espera que tras aplicar el uso de drones en carreteras se reduzcan de manera considerable este tipo de incidencias. Además, mediante este método podría cubrirse una cantidad de terreno mucho mayor a si se usa un helicóptero o un coche policial básico.

Según un informe de la DGT emitido el 7 de julio de 2021 en el que se muestra los drones presentes en todo el territorio español, como se muestra en la Figura 27, han estado en vuelo unas 500 horas desde 2018 y han detectado más de 600 infracciones, habiéndose convertido en uno de los métodos más efectivos de infracciones como el uso del móvil al volante, uso incorrecto del cinturón y de los sistemas de retención infantil, además de adelantamientos sin respetar la distancia a ciclistas. Por lo que, se puede suponer que han sido una gran adición a la Dirección General de Tráfico. [52]

Esto, obviamente, es en el entorno real, en nuestro proyecto se debía hacer en un entorno simulado por los problemas de la legislación con el vuelo comentados en el apartado anterior y con problemas surgidos con la privacidad.



Figura 27. Distribución de drones en España [52]

5.2 Impacto Económico

Otro apartado donde puede ser interesante aplicar este proyecto en un entorno real sería desde la mirada económica, puesto que actualmente lo más parecido a este método serían los helicópteros y estos no son especialmente baratos además de no contar con las potentes cámaras de los drones modernos que pueden tener visión térmica, un potente zoom y más que hacen que en este aspecto el dron salga ganando.

Según un artículo de Lisa Institute [53], usar drones en situaciones de emergencia es 12 veces más rentable que usar los helicópteros y aviones tripulados convencionales. Por lo que es una ventaja si se compara con otros vehículos aéreos que realizan trabajos de monitorización de carreteras.

5.3 Impacto Medioambiental

Al ser vehículos que no funcionan por gasolina y funcionan mediante baterías, la energía necesaria para usarlos podría proceder de otros medios distintos al petróleo menos contaminantes mediante las energías renovables como placas solares o energía eólica, siendo esta otra posible ventaja respecto a los helicópteros.

Si bien es cierto que las baterías están construidas normalmente de litio y este no es un material especialmente sano para el medio ambiente, es una solución mejor que la utilización de compuestos derivados del petróleo y si se mejorase estas energías renovables para obtener una mayor cantidad o se apostase más por estas sería incluso más óptimo si cabe.

Según un artículo de El Confidencial [54], un estudio de Volvo ha concluido que la fabricación de un coche eléctrico es mucho más perjudicial para el medioambiente por su costo en emisiones de CO₂ en la fabricación de las baterías de ion de litio que contamina más que la fabricación de motores normales a base de gasolina, pero también detalla que, a lo largo del tiempo, esta deficiencia en cuanto a las emisiones se reduce y acaba reduciendo sus emisiones con respecto a su contrapartida de gasolina. Aunque habría que ver de qué forma se consigue la energía y si los métodos utilizan un porcentaje alto de energías no renovables porque si no la diferencia no sería un detalle para destacar.

Aunque el artículo anterior relata la diferencia en coches, el costo de fabricación de baterías para drones sería el mismo de manera porcentual, a más pequeña la batería menos CO₂ se emite, pero sería el mismo, y encima se está comparando con otras alternativas como el helicóptero cuya emisión en su fabricación es bastante mayor que un dron si obviamos la batería de ion litio.

Por lo que, para acabar, si bien el dron a corto plazo es perjudicial para el medioambiente, la reutilización de estas baterías y su vida útil hacen que sea una posibilidad más beneficiosa para el medioambiente que los compuestos derivados del petróleo.

6 DISEÑO DE LA SOLUCIÓN

Como consecuencia de todo lo explicado en el documento anteriormente, se ha conseguido el desarrollo de una aplicación funcional que va a ser detallada en este apartado. Para ello, se hará un análisis de los requisitos que necesita el sistema, con todo lo que ello conlleva, además de explicar cómo funciona el sistema y todas sus partes, haciendo hincapié en los algoritmos y funcionalidades que este utiliza o que se han desarrollado para el mismo.

El desarrollo de un proyecto es la parte más importante de una memoria relacionado del mismo, si bien otros apartados también son importantes sin este apartado no hay proyecto, por eso es necesario un análisis de cómo se ha hecho y por qué de esa forma.

6.1 Tecnologías empleadas

A pesar de que ya se ha comentado más o menos anteriormente en el apartado de estado del arte, se va a hacer una recapitulación de las herramientas y tecnologías empleadas en el desarrollo del sistema y por qué de su uso en el mismo. Entre ellas encontramos:

- Visual Studio y Visual Studio Code: Para la realización de todo el código tanto de Python que implica las acciones del dron, como en la parte de desarrollo de un algoritmo de detección y tracking de los vehículos. Visual Studio ha sido utilizado a la hora de realizar cualquier código necesario para el Unreal Engine, como nuevas blueprints, y esto será explicado más detalladamente más adelante.
- Python y C++: Aunque C++ solo ha sido utilizado para la creación de unas blueprints personalizadas cabe destacarlo como lenguaje de programación utilizado en la realización del sistema, una gran mayoría del mismo ha sido desarrollado mediante Python.
- AirSim, PX4 y herramientas de control del dron como MAVLink o MavSDK se han utilizado para conseguir una comunicación del dron con el código realizado, aportarle realismo a la simulación y conseguir que las acciones del dron fueran llevadas a cabo.
- Unreal Engine y Cesium: Se ha utilizado el motor gráfico Unreal Engine para realizar estas simulaciones llevadas a cabo en el sistema y para aportar más realismo y que no fuera en un mundo plano o creado a través del propio motor, se ha utilizado la herramienta de Cesium para tener a mano mapas del mundo real.
- OpenCV: Para conseguir los algoritmos relacionados con la visión por ordenador y tener un buen tracking y detección de los vehículos para posteriormente calcular la velocidad se ha utilizado OpenCV.
- Google Maps: Como bien se ha comentado anteriormente, para calcular la velocidad de una manera más o menos fiable, se necesita tener información del tamaño de la carretera mostrado en el vídeo y para ello se usa Google Maps.

6.2 Identificación de requisitos

Para mostrar de una forma clara todas las funcionalidades del sistema y relacionar cada una de las funcionalidades y características del sistema con distintos valores a presentar.

Los requisitos que van a ser descritos en este apartado son únicamente del sistema puesto que el proyecto que se ha realizado no está enfocado a un público para ser usado y, por tanto, no es posible realizar estos requisitos desde la posición de un usuario.

Como solo van a existir requisitos del sistema, estos se dividen en:

- **Requisitos funcionales:** Se refiere a cualquier acción que puede llevar a cabo el sistema y como las realiza para cumplir los objetivos del proyecto.
- **Requisitos no funcionales:** Reflejan los límites presentes dentro del sistema, todas las métricas de rendimiento que muestra el mismo y el entorno y herramientas que deben de ser usadas.

Los atributos que presentará cada una de las tablas de los requisitos del sistema son los siguientes:

- **ID:** El ID del requisito en cuestión, seguirán el siguiente formato XX-YY donde XX será RF o RN dependiendo de si es funcional (RF) o no funcional (RN) e YY será el número del requisito (Ej: 01).
- **Título:** El nombre del requisito, debe ser breve y conciso.
- **Tipo:** Refleja si es funcional o no funcional.
- **Prioridad:** La prioridad del requisito, esta puede ser Alta, Media o Baja dependiendo de lo importante que es dentro del sistema.
- **Verificabilidad:** Si es posible ser verificado o no mediante casos de prueba, comprobación directa o técnicas para comprobar su correcto funcionamiento. Es un atributo con dos únicos valores: SÍ o NO.
- **Descripción:** Es la descripción del requisito, donde se explica a qué se refiere y que denota el requisito.

A continuación, se muestra un ejemplo de requisito utilizando el formato ya comentado:

ID	RX-YY				
Título	Ejemplo de requisito				
Tipo	Funcional o No funcional	Prioridad	Alta Media Baja	Verificabilidad	Sí No
Descripción	Es el ejemplo a seguir para todos los requisitos siguientes.				

Tabla 8. Formato base de los requisitos

6.3 Casos de uso

Continuando con los métodos para mostrar si las funcionalidades del sistema son válidas, los casos de uso son un método para obtener requisitos del sistema. Estos detallan las acciones que debe tomar el sistema para realizar un proceso específico.

Por lo tanto, se van a describir varios casos de uso generales para que posteriormente se validen los requisitos mediante la matriz de trazabilidad. No son casos de uso específicos ni pruebas del sistema, que se mostrarán más adelante en el apartado de evaluación, así que no se hará mención a la cantidad de coches, velocidad específica ni nada muy concreto.

El modelo que seguirá cada caso de uso es el mostrado a continuación:

ID	CU-XX
Nombre	Ejemplo de caso de uso.
Descripción	Es el modelo de tabla para los casos de uso.
Precondición	
Postcondición	
Acciones	

Tabla 9. Modelo de tabla de los casos de uso

Los campos de los casos de usos son los siguientes:

- **ID:** Describe el identificador del caso de uso siguiendo el siguiente modelo CU-XX.
- **Nombre:** Es el nombre escueto y que identifica el caso de uso.
- **Descripción:** Describe el caso de uso de manera más concreta.
- **Precondición:** Son las condiciones que debe tener el sistema para realizar la acción.
- **Postcondición:** Cómo se encuentra el sistema una vez realizada la acción.
- **Acciones:** El conjunto de acciones que debe seguir el sistema para cumplir el caso de uso.

Por su elevada cantidad la lista de requisitos se encuentra disponible en el Anexo B.

ID	CU-01
Nombre	Seguimiento de un camino por un coche.
Descripción	Un coche deberá seguir un camino preestablecido a una velocidad determinada.
Precondición	Tener un componente ‘Spline’, que representa el camino, y un actor de un coche.
Postcondición	El o los coches se moverán siguiendo los puntos establecidos a una velocidad constante.
Acciones	<ul style="list-style-type: none"> • Colocar uno o más vehículos en el mapa. • Colocar el componente ‘Spline’. • Establecer los puntos del camino que deben seguir los coches. • Establecer la velocidad a la que se moverán los coches.

Tabla 10. Caso de uso 01

ID	CU-02
Nombre	Grabación del dron y movimiento del coche.
Descripción	El dron se moverá a una ubicación mirando una carretera y grabará un vídeo del tiempo que se quiera mientras los coches se mueven.
Precondición	Tener un dron en una ubicación del mapa y estar simulando mediante ‘AirSimGameMode’ en un entorno real de Cesium y uno o varios coches en la carretera objetivo.
Postcondición	Se obtendrá un vídeo mediante la cámara del dron de uno o varios coches moviéndose a través de una carretera.
Acciones	<ul style="list-style-type: none"> • El dron se conectará al computador. • El dron se desplazará a donde se indique. • El coche espera el tiempo requerido. • Colocar la cámara mirando a la carretera. • El coche comienza su recorrido. • Grabar un vídeo del tiempo requerido.

Tabla 11. Caso de uso 02

ID	CU-03
Nombre	Procesado de vídeos.
Descripción	Se tomará un vídeo y se procesará para detectar vehículos, realizar su tracking y estimar la velocidad.

Precondición	Poseer un vídeo de una carretera con vehículos.
Postcondición	El vídeo será procesado aportando información del vídeo y mediante ella hacer una estimación de la velocidad.
Acciones	<ul style="list-style-type: none"> • Se procesa el vídeo específico. • Se obtiene información a partir del vídeo. • Se estima la velocidad mediante esa información. • Se compara y evalúa con la velocidad real del coche.

Tabla 12. Caso de uso 03

6.3.1 Matriz de trazabilidad

Para mostrar la relación que tiene cada uno de los requisitos funcionales del sistema entre ellos va a realizarse una matriz de trazabilidad dejando claro que requisitos están relacionados con que casos de uso.

Caso de uso Requisito	CU-01	CU-02	CU-03
RF-01		X	
RF-02		X	
RF-03		X	
RF-04		X	
RF-05			X
RF-06	X	X	
RF-07	X	X	
RF-08	X	X	
RF-09	X	X	
RF-10	X	X	
RF-11	X	X	
RF-12		X	X
RF-13		X	X
RF-14		X	X
RF-15		X	X
RF-16		X	X
RF-17		X	X
RF-18		X	X
RF-19			X
RF-20			X
RF-21			X

Tabla 13. Matriz de trazabilidad de los requisitos funcionales vs Casos de uso

6.4 Arquitectura del sistema

El sistema tiene unos componentes que están interrelacionados para su funcionamiento y que toman los datos o funcionalidades que realizan otras partes del mismo para realizar sus propias acciones. En este apartado se va a explicar cada uno de los componentes del sistema y como se relacionan entre ellos.

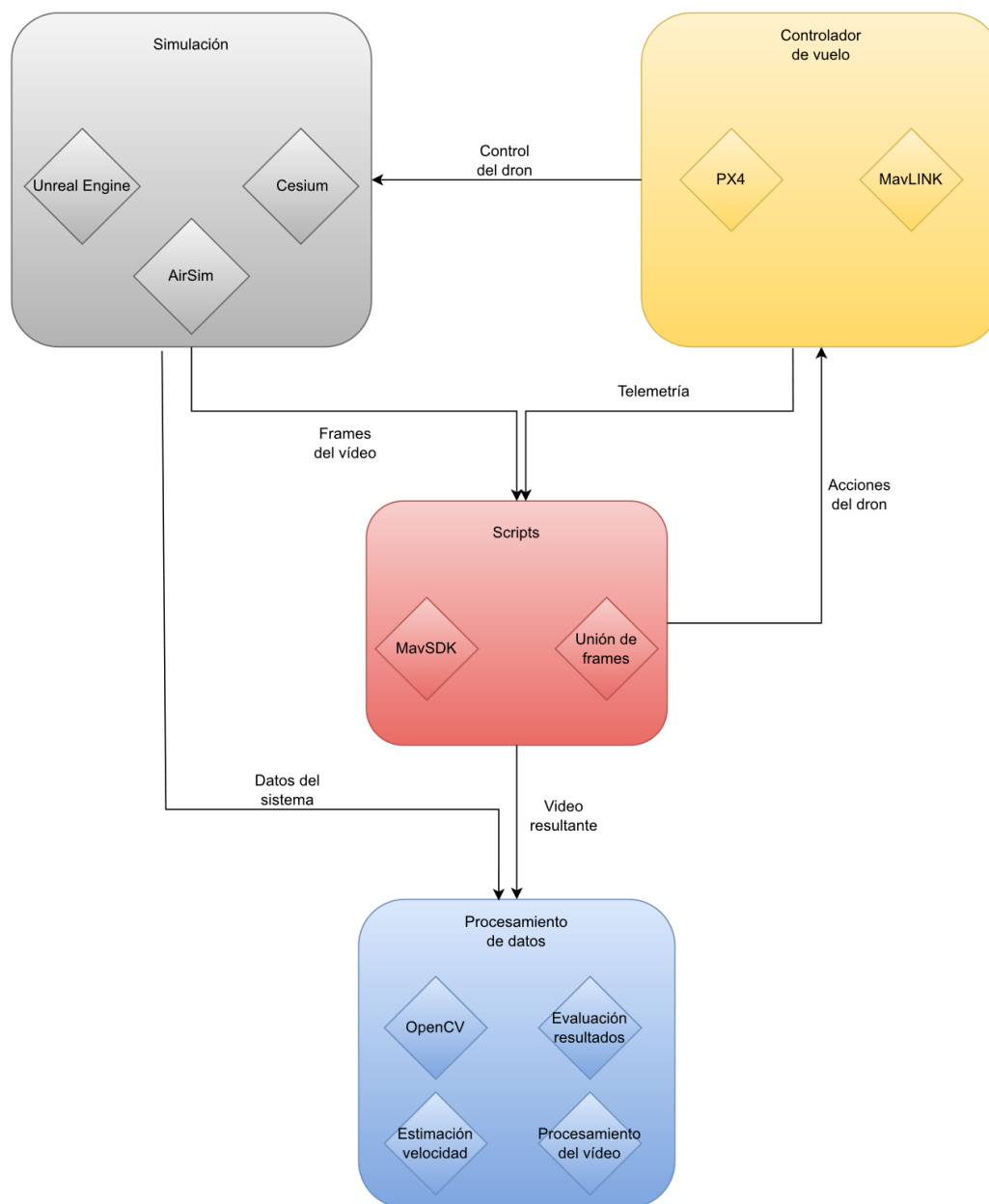


Figura 28. Arquitectura del sistema

Como se puede ver en la imagen anterior, el sistema posee cuatro componentes generales. En ellos se realizan distintas acciones y tienen salidas de información que usan otros componentes del sistema. Los componentes son los siguientes:

- **Simulación:** Se refiere al apartado del sistema necesario para simular la misión del dron, desde el Unreal Engine, el propio plugin de este para aportar realismo Cesium y el simulador utilizado como tal: AirSim.
- **Controlador de vuelo:** Es toda la comunicación entre el sistema y el dron y el que hace que las acciones que se quieran tomar se realicen correctamente en el simulador.
- **Scripts:** Si bien en el procesamiento de datos también se podría considerar script, se ha decidido que forma parte del componente final por obtener los datos definitivos en él, son tanto los scripts de movimiento del dron, posición de la cámara y el código que une los frames conseguidos en el simulador al grabar.
- **Procesamiento de datos:** Toma tanto el vídeo resultante del dron como los datos recogidos del sistema, como la posición del coche, su velocidad, un timestamp, etc., para procesar el vídeo y realizar su tracking. A partir de esto, se puede estimar la velocidad y comparar con los datos reales del sistema.

6.4.1 Conexión del sistema e inicialización del mismo

Aunque se podría considerar como la conexión del componente de scripts con el simulador mediante PX4, cabe recalcar como se conecta el sistema con el computador para que la simulación se realice correctamente y como se inicia este.

Se ha desarrollado un archivo .bat que inicia todo el sistema, abre WSL con todas las carpetas del sistema, abre Unreal Engine y modifica las IPs para que se conecte correctamente con PX4.

Una vez abierto todo el sistema, se necesitará que en ejecución el dron esté conectado al computador para ello se han debido crear un par de reglas tanto de entrada como de salida para permitir que PX4 y MAVLink utilicen una serie de puertos para esta conexión.



Figura 29. Reglas de entrada y salida

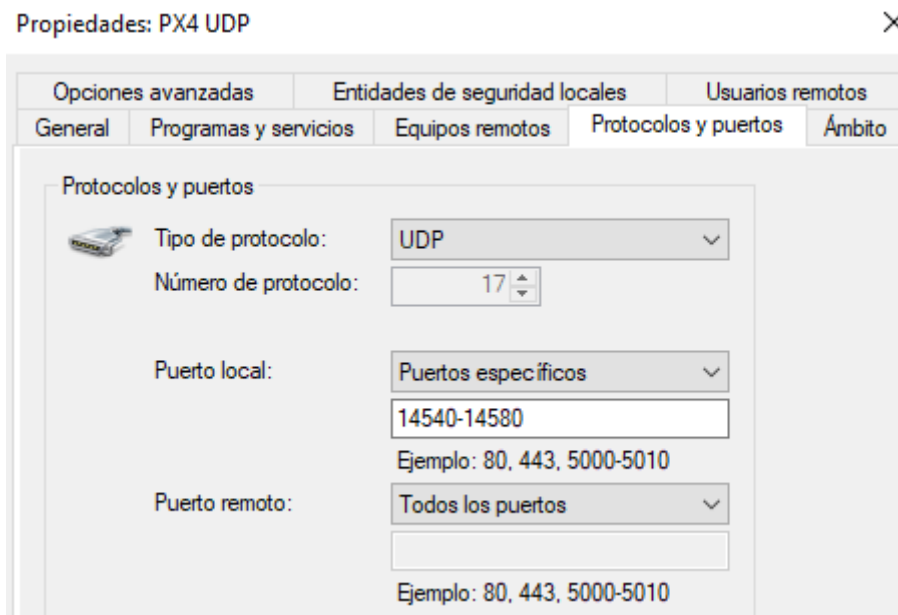


Figura 30. Regla UDP

En específico es importante las reglas UDP, que en este caso abren los puertos del 14540 al 14580 para su conexión con el sistema, puesto que a la hora de realizar la conexión del dron con el computador mediante la ejecución de un archivo .sh se utiliza el siguiente comando: **./local/lib/python3.8/site-packages/mavsdk/bin/mavsdk_server udp://:14551 -p 50051** en el final del archivo y que realiza la conexión mediante MAVLink con el propio dron y utiliza el puerto 14551 con UDP para este y como se puede observar entra dentro del rango que se ha señalado.

Al conectar todo esto, el sistema está listo para cualquier acción que requiera el dron y esté conectado al computador.

6.5 Diseño de la solución

En este apartado se va a proceder a realizar una explicación de todas las partes presentes en el sistema utilizando diagramas de flujo e imágenes, aunque no del sistema final, para ello.

Como el sistema se divide en varios apartados, movimiento en patrulla de los vehículos o el propio algoritmo de visión por ordenador, se va a dividir la explicación del diseño en estas partes.

6.5.1 Diseño de la misión del dron

Para la realización de la misión del dron se debe tener en cuenta el punto de origen de coordenadas, que se encuentra en coordenadas GPS, y a partir de este se ha utilizado las coordenadas NEDYaw, que hace referencia a cuanto se mueve al Norte (North), Este

(East) y hacia abajo (Down) y cuánto gira en cuánto a su dirección. Mediante esto, se puede hacer que se eleve para evitar posibles obstáculos y desplazarse a cualquier lugar del entorno simulado, aunque para que se desplace de forma correcta y no empalme un desplazamiento o movimiento con otro se debe realizar esperas de unos segundos para que el dron pueda realizarlo.

Una vez realizado el movimiento a la zona deseada y configurada la cámara en el ángulo requerido se dispondrá a realizar la grabación de un tramo de la carretera mediante la función de grabación del dron y se detendrá la misma cuando se considere necesario.

En el siguiente diagrama de flujo se explica de forma visual:

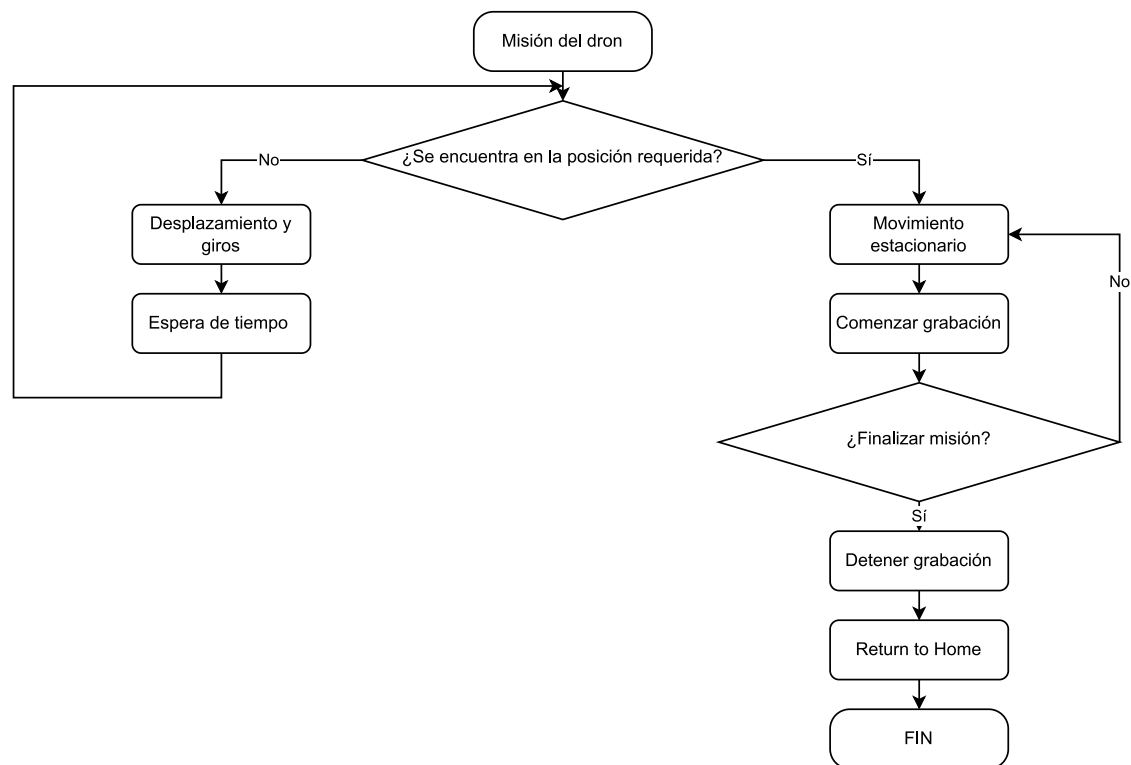


Figura 31. Diagrama de flujo misión del dron

6.5.2 Diseño de las tareas de los coches

Para la realización del sistema es necesario que al menos los coches se muevan en una dirección y así poder hacer una estimación de la velocidad del vehículo.

Para la consecución de esta funcionalidad se ha seguido el modelo de patrulla básica de los videojuegos donde se establece un camino antes de la ejecución mediante la creación de un actor de Unreal Engine, que en nuestro caso se ha llamado VehiclePath, y añadiendo una componente 'spline' al mismo para que funcione como puntos de un camino a seguir, como se puede ver en la Figura 32.

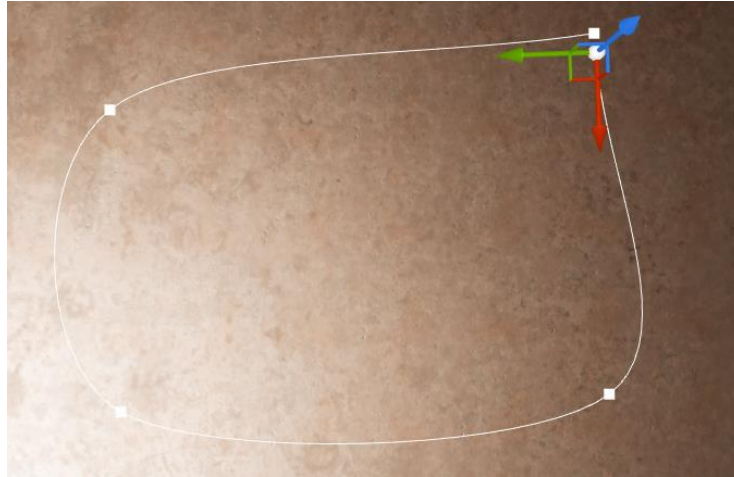


Figura 32. Ejemplo camino Spline

Con estos puntos preestablecidos, se es posible generar cualquier tipo de camino y que el coche los recorra. Los recorre haciendo uso de la función que se ha creado llamada *GetPath*, que es una función que toma el array de puntos del camino y le va dando la dirección de estos al vehículo para que se dirija a suposición, todo esto dentro del evento ‘Tick’ que se ejecuta en cada frame del juego.

La función *GetPath* realiza una serie de cálculos matemáticos para saber la posición de cada punto respecto a donde se encuentra el propio coche. Estos cálculos son los siguientes: Primero toma el array del camino y devuelve el primer elemento del mismo haciendo una copia de este, cuando pasa por este punto el camino pasará a no estar el primero en la lista, y mediante este elemento, que es una localización dentro del mundo, se consigue el vector tangente más cercano a esta posición para normalizar este vector y escalarlo posteriormente con una tolerancia de 0,0001, es decir puede variar el resultado del escalado del vector ese valor. Con el vector ya escalado se suma con el vector obtenido de la posición del vehículo, que se obtiene mediante la función *GetActorLocation* y como el actor en este caso es el propio vehículo te devuelve su posición en forma de vector. Este vector sumado se utiliza para obtener el punto de la curva más cercano a esta ubicación, que será tomado como el punto a avanzar, y tomando la posición del vehículo y esta posición meta se consigue la rotación que se debe realizar hacia ese punto, si se junta con la función *GetActorRotation*, que toma la rotación actual del vehículo, al normalizarlo con estos dos vectores y la función delta de Unreal Engine de la rotación se consigue la dirección a donde se tiene que mover (Yaw) y se escala este ángulo de rotación respecto a 90 y -90 tomando desde 1.0 y -1.0, es decir en un punto inicial de rotación 0.5 al pasarlo por esta función el nuevo valor sería de 45, y se pasa por fin como parámetro de salida para que tome esa dirección en concreto.

Para concluir con las funcionalidades del movimiento del coche, se ha desarrollado una funcionalidad para mantener una velocidad que se proponga a lo largo del tiempo, para poder tener otro método más de comprobar la velocidad a parte del CSV que se escribe y, además así se comprueba que esta velocidad del archivo del log es correcta. La velocidad se calcula mediante el uso de la función *GetVelocity* en Unreal Engine que te

lo devuelve cm/s y luego mediante unos cálculos pasarlo a km/h en el vector XY para saber la velocidad correcta.

Esta velocidad en km/h se puede pasar de un vector general al XY de dos maneras, que han sido implementadas ambas para comprobar su funcionamiento. La primera consiste en comprobar la longitud del vector, según sea el vector de velocidad más largo tendrá una mayor velocidad, y multiplicarlo por 0,036 para pasarlo a km/h y el otro método es el de utilizar la siguiente ecuación: $\sqrt{x^2 + y^2}$, siendo x e y los dos vectores de velocidad en esos planos, para obtener el valor del vector en XY y de nuevo se pasa a km/h.

Entonces una vez se tiene estas dos cosas se compara la propia velocidad del coche con una que se establezca como límite y cuando se encuentre por debajo se apague al freno poniéndolo a 0 y estableciendo la aceleración a 1.0 (la máxima posible) y si lo supera viceversa.

Otra funcionalidad que presentan los vehículos es la escritura de información del mismo en tiempo de ejecución dentro de un archivo CSV. Esta información es la siguiente: Posición del coche en X, Y y Z con respecto a la posición del dron porque para calcular las coordenadas en el mundo real se debería tener estas coordenadas, el timestamp de cuando se ha tomado la información, el tag del vehículo, la posición del dron y la velocidad del coche en todos los planos (X, Y, Z, XY).

Se utiliza una 'blueprint' personalizada, ya que dentro del propio Unreal Engine no existe una forma de escribir en un archivo CSV y se ha tenido que implementar en C++. Para llamar a esta función se ha usado un contador cíclico que se ejecuta cada segundo (o el tiempo que se considere necesario) y se va escribiendo en cada línea del archivo.

Para terminar con las funcionalidades de los coches, el coche es capaz de esperar un número determinado de tiempo antes de que se disponga a avanzar o recoger información para que a la hora de realizar las pruebas al dron le dé tiempo a llegar al lugar específico antes de que avance el coche.

Como la función GetPath posee un gran número de cálculos matemáticos mediante vectores y demás y que estos han sido explicados anteriormente, a la hora de realizar el diagrama de flujo que explica la funcionalidad de movimiento no se explicarán y aparecerán como 'Aplica cálculos matemáticos'. El siguiente diagrama de flujo explica de manera visual todas las funcionalidades de cada coche explicadas anteriormente:

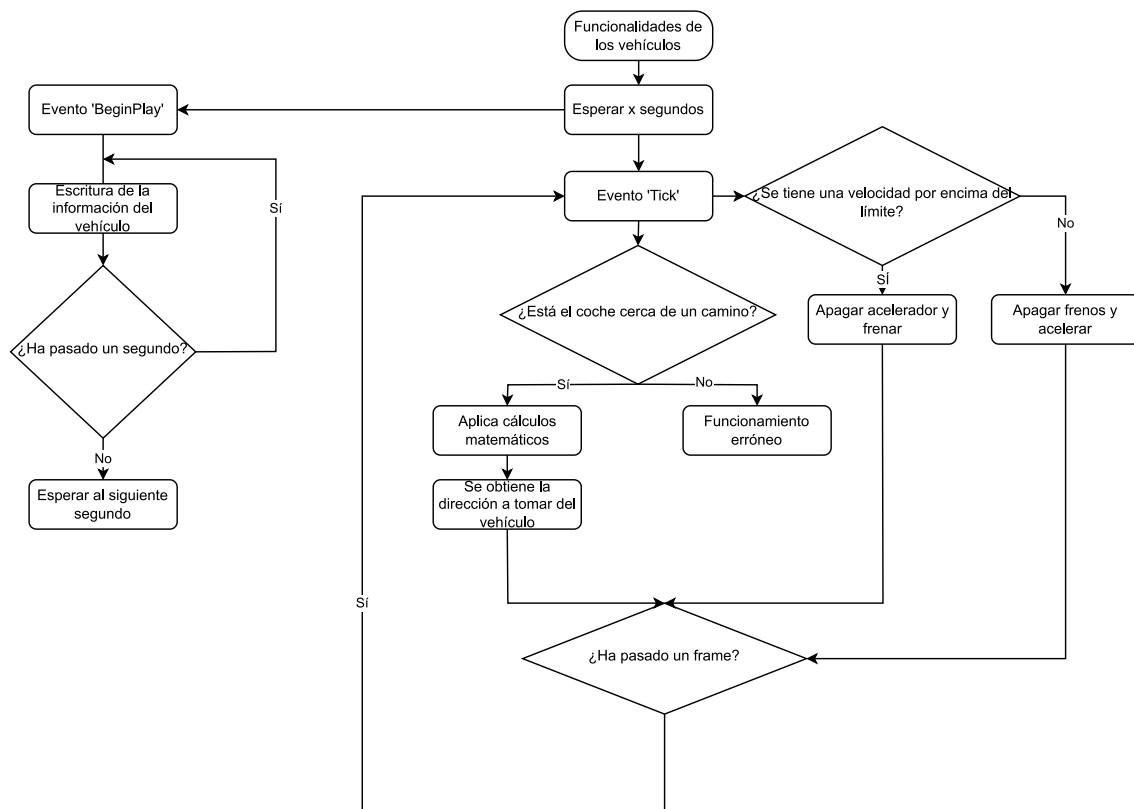


Figura 33. Diagrama de flujo de las tareas de los vehículos

6.5.3 Diseño del algoritmo de estimación de velocidad

Una vez se tiene ya todos estos componentes funcionando, se tomará el vídeo resultante de la grabación y se aplicará un algoritmo de visión por computador para la detección de los vehículos, su tracking y una estimación de la velocidad a la que se están moviendo. Antes de proceder, cabe resaltar que cada vez que se mencionen funciones son funciones del propio OpenCV. Y también es bueno señalar que se realiza un cambio de resolución del vídeo para poder comprobar las tres ventanas resultantes al mismo tiempo (máscara, región de interés y vídeo).

Para la detección se ha utilizado un sustractor del entorno que usa el algoritmo KNN explicado en el apartado del estado del arte, y mediante el detector resultante de este se computa una máscara del entorno solo sobre la región de interés, que en nuestro caso es la carretera puesto que se obvian los árboles y demás. A partir de esta máscara se aplica la función threshold que binariza la imagen y mediante la función findContours se encuentran los contornos de los objetos en esta imagen binarizada. Recorriendo todos los contornos del frame, se consigue el área del contorno mediante la función contourArea y dependiendo de si el contorno supera un área especificada será considerado un objeto y se guardará su bounding box mediante el uso de la función boundingRect, que devuelve

el punto (x, y) superior izquierdo del cuadrado, su anchura y su altura, para posteriormente guardar esta bounding box en una lista.

En el siguiente diagrama de flujo se puede observar de manera más visual el apartado de detección de los vehículos:

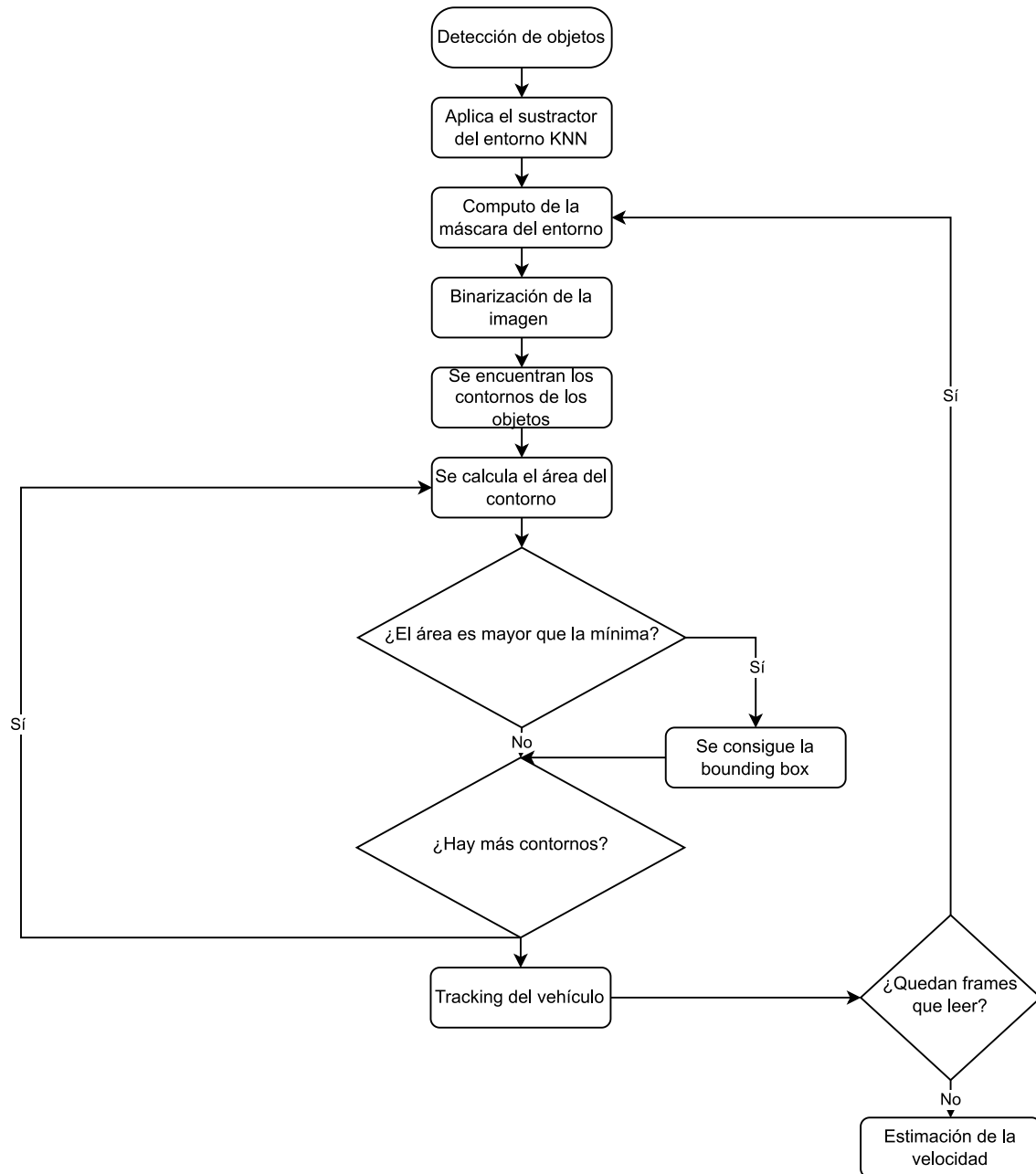


Figura 34. Diagrama de flujo detección de los vehículos

En cuanto al tracking de los vehículos, se toma todas las bounding boxes obtenidas mediante la detección, se calcula el centroide del rectángulo. Esto es, mediante el siguiente cálculo: $\frac{(x+x+w)}{2}$ para el plano X y $\frac{(y+y+h)}{2}$ para el plano Y, siendo las dos fracciones funciones suelo.

Si ese objeto no ha sido detectado, que se maneja con un booleano, se establece como un objeto de Python y se añade a una lista donde se encontrará la bounding box y la id de cada objeto, además de sumarle uno a los ids de los objetos para tener constancia de los que ha habido. Una vez detectado el objeto se comprueba si la distancia euclídea, que se calcula con la siguiente formula: $\sqrt{(x - x_{Anterior})^2 + (y - y_{Anterior})^2}$ entre el actual centroide y el anterior supera un umbral específico, si no lo supera se considerará como el mismo objeto y se añadirá a la lista comentada anteriormente, con esto se escribirá en el CSV la siguiente información cada vez que no supere el umbral: el id del objeto, su centroide en el frame, un timestamp y si ha superado un punto de la imagen que se explicará más adelante su uso.

Cuando haya terminado este proceso, se irá recorriendo la lista de bounding boxes e ids y se mostrará en la imagen la bounding box en la posición requerida y un texto con el siguiente formato: Obj: ID del objeto.

En el siguiente diagrama de flujo se puede observar de forma más simple lo explicado recientemente:

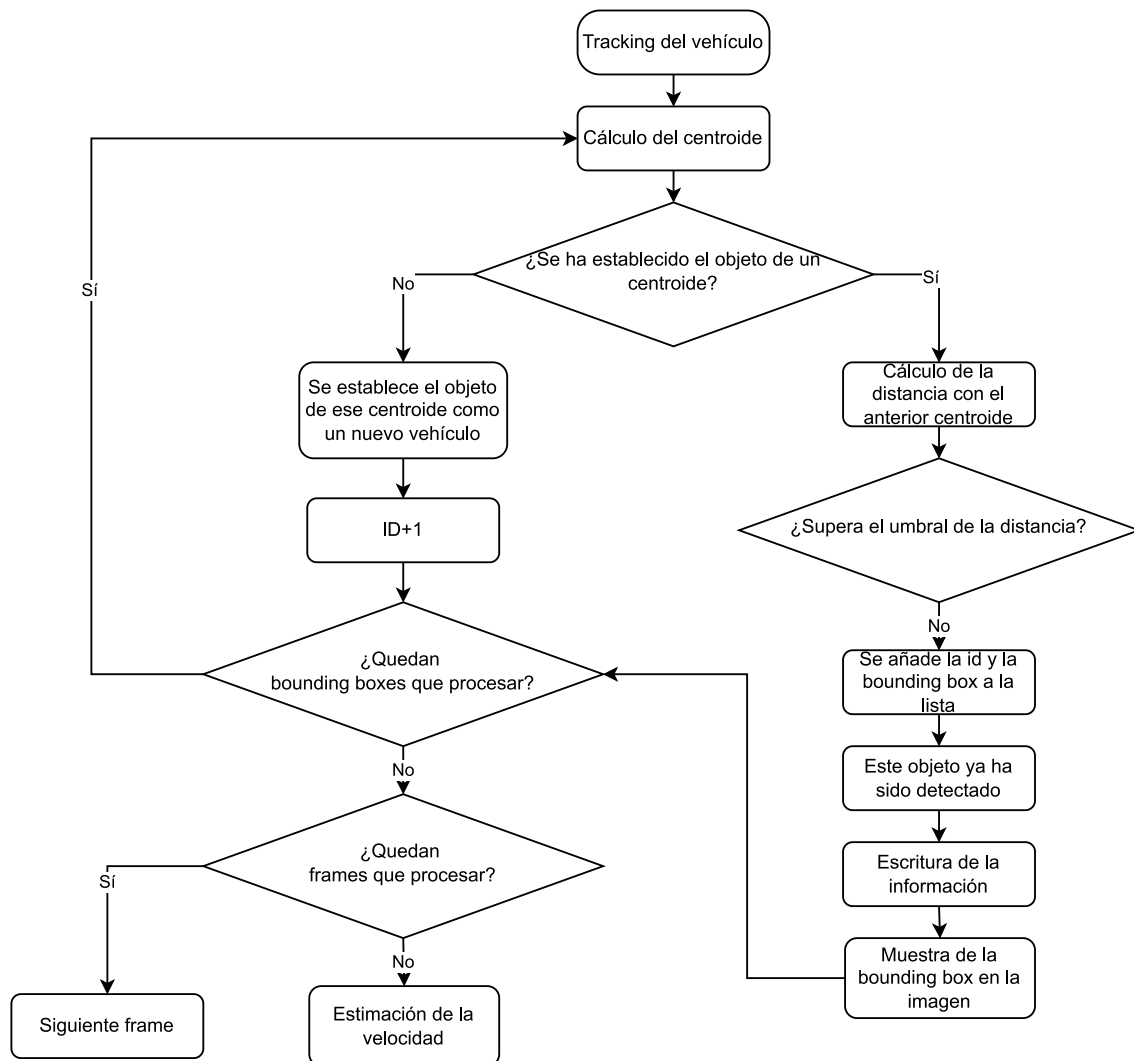


Figura 35. Diagrama de flujo del tracking de los vehículos

Y, para terminar con este apartado, quedaría explicar cómo se ha realizado la estimación de la velocidad a partir de la información obtenida en el procesado de la imagen.

La velocidad se ha calculado con la fórmula física $velocidad = espacio/tiempo$ donde el espacio es una distancia estimada a través de elementos de la imagen como un banco de Cesium, que, aunque está aplastado se puede usar para saber dónde se encuentra utilizando Google Maps, o las señales de tráfico que estas si están en 3D. A partir de esto, se puede obtener una distancia estimada más o menos útil para poder usarla en el cálculo. El tiempo que se utiliza en la fórmula es la diferencia entre el primer tiempo en el que se reconoce el objeto en el procesado de vídeo y la primera vez que se encuentre un 1 en el CSV asociada al ID de ese objeto, puesto que eso significará que el centroide ha pasado por la línea utilizada en el vídeo que emula la distancia estimada anteriormente. Mediante esto se calcula la velocidad a la que va el vehículo.

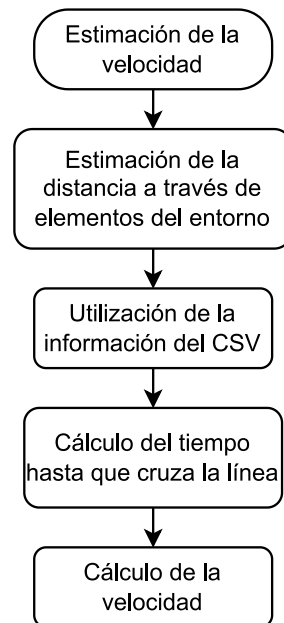


Figura 36. Diagrama de flujo de la estimación de la velocidad

6.6 Alternativas a la solución / Alcance de la solución

Si bien se ha alcanzado una solución funcional para el sistema, para llegar a esta se han probado otras alternativas y pruebas que no han dado los resultados esperados. Es decir, se va a comentar que se ha probado dentro del sistema y a donde ha llegado este.

Una de las alternativas que se ha probado ha sido la utilización de YOLO, un software capaz de facilitar el reconocimiento de vehículos y realizar su trackeo de manera más sencilla y, si bien su funcionamiento era óptimo en pruebas de vídeos del mundo real, todos los algoritmos que se han intentado con este al pasarlos al entorno simulado han resultado en un intento fallido al colocar bounding boxes en lugares donde no hay ni vehículos ni objetos de interés, fallos a la hora del tracking de los mismos y por ello se ha descartado esta propuesta. Aunque, cabe recalcar que YOLO es un software que presenta muy buenos resultados en general pero que en este caso en sí no se ha podido llevarlos al entorno simulado.

También se probó un software con unos pesos entrenados de un dataset de más de 200.000 imágenes de vehículos de China [44] y aun así no consiguió resultados esperados lo suficientemente buenos.

Dentro de la propia solución del sistema para la detección de vehículos, que utiliza un sustractor del entorno basado en el algoritmo KNN, se probó una alternativa basada en una mezcla gaussiana para comprobar si detectaba los objetos de interés de mejor manera y se llegó a la conclusión de que el algoritmo KNN era el más óptimo de ambos.

Además, se ha probado el método para binarizar cada frame utilizando Otsu para ello, dando resultados, si bien óptimos, algo peores a comparación que una binarización común.

En otros apartados del sistema, como la unión de los frames en un vídeo se probó dos alternativas, una en la que se accedía a cada imagen a través de la librería `os` y otro mediante `glob` (la unión de los frames se sigue realizando mediante `OpenCV`), tenía una velocidad algo más rápida mediante la librería `glob` para acceder a las imágenes, además de un código algo más simple y por esto se decidió usar la librería `glob` para esto.

Centrándose en el alcance de la solución, se ha conseguido tener una estimación, si bien no es exacta, de la velocidad, pero en los inicios del proyecto se intentó reconocer las líneas de la carretera para poder captar infracciones como adelantamiento por línea continua y no se consiguió obtener un resultado satisfactorio del mismo debido a cómo funciona la cámara del sistema y la complejidad de esta funcionalidad y, por tanto, se decidió centrarse únicamente en la estimación de la velocidad. También cabe señalar que se intentó tener otro método para obtener la distancia, haciendo cálculos para obtener las coordenadas en tiempo real del dron y del coche respecto al dron, pero no eran del todo exactas dentro de `Cesium` y surgía el mismo problema que para conocer el FOV (Field of view) del dron, tenía que ser estimada también, así que se decidió continuar con el método de estimación de la distancia a través de los elementos del entorno visibles en la cámara.

7 EVALUACIÓN DE LA SOLUCIÓN

Teniendo en cuenta el apartado anterior, se han desarrollado una batería de pruebas para comprobar el correcto funcionamiento del sistema y realizar un análisis de los resultados obtenidos en el sistema. Para ello, se harán pruebas de los distintos componentes del sistema desde la propia cámara, dejando ver que funciona de manera correcta, hasta el propio algoritmo haciendo pruebas con distinto número de vehículos, tomando como referencia los casos de uso enunciados anteriormente.

Pero, antes de empezar la evaluación de estas pruebas, hay que indicar el estado inicial del sistema y desde dónde empieza el dron. Para las pruebas se ha usado el mapa de Cesium del campus de Colmenarejo de la Universidad Carlos III de Madrid, siendo la posición inicial la siguiente: (Latitud: 40.544289, Longitud: -4.012101), que es la que se presenta en la Figura 37 y es donde el dron empieza a realizar su movimiento trasladando a la carretera de la izquierda para comenzar la grabación.

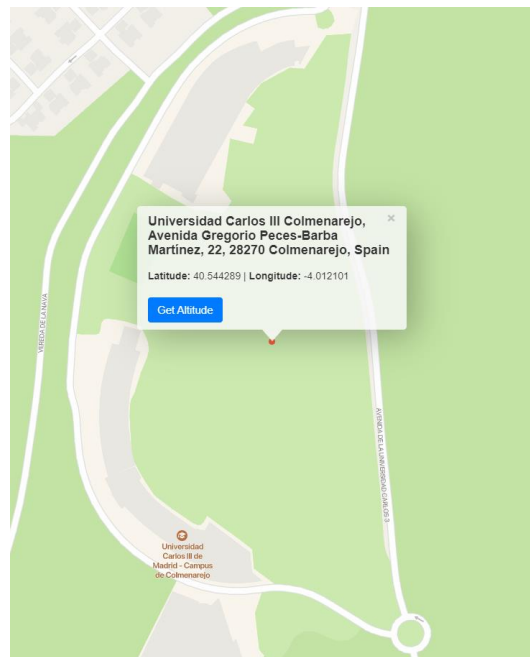


Figura 37. Ubicación inicial del dron

7.1 Pruebas de la misión autónoma

Como se ha explicado antes, la misión autónoma básicamente consiste en el movimiento a un punto arbitrario del mapa usando coordenadas NEDYaw, que funcionan como ya se ha explicado anteriormente, y comenzar a grabar cuando se desee.

Una vez comience la ejecución se debe poner la posición inicial como la posición (0.0, 0.0, 0.0, 0.0) para tomar este como el punto de referencia y una vez establecido este, el dron tomará vuelo y se desplazará a donde se le indique, en nuestro caso se hacen movimiento para llegar a la siguiente ubicación:

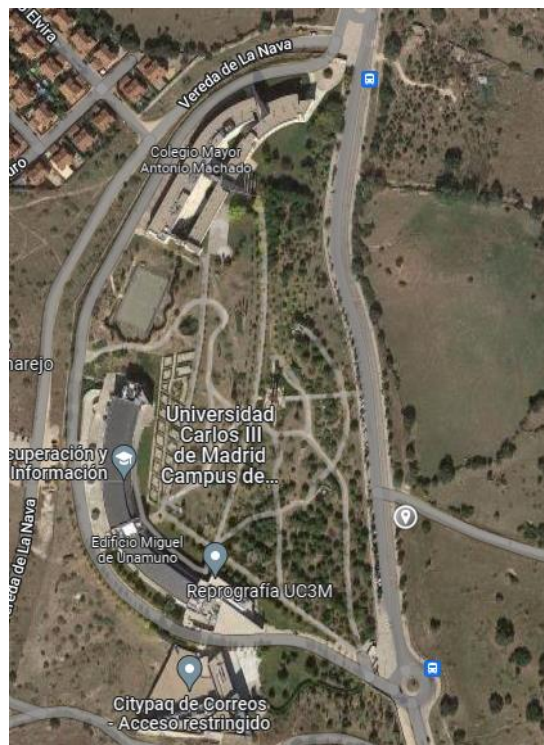


Figura 38. Ubicación destino del dron

Entonces, comprobando que se mueve al punto requerido, el dron se coloca en la ubicación mostrada en la Figura 39, aunque si bien no se aprecia el dron se encuentra ahí, pero se ha alejado la cámara para obtener una vista desde arriba.



Figura 39. Ubicación destino dentro del simulador.

Como se ve en la imagen, se desplaza a la ubicación que se proponía en un primer momento.

Una vez llegado a este punto enfocará a la carretera más cercana que tiene y empezará a grabar, sacándolo como un conjunto de frames, se tomará uno de estos frames para resaltar el correcto funcionamiento de la cámara y teniendo en cuenta que se ha establecido una resolución de 1920x1080, como se ve en la siguiente imagen funciona de manera adecuada:

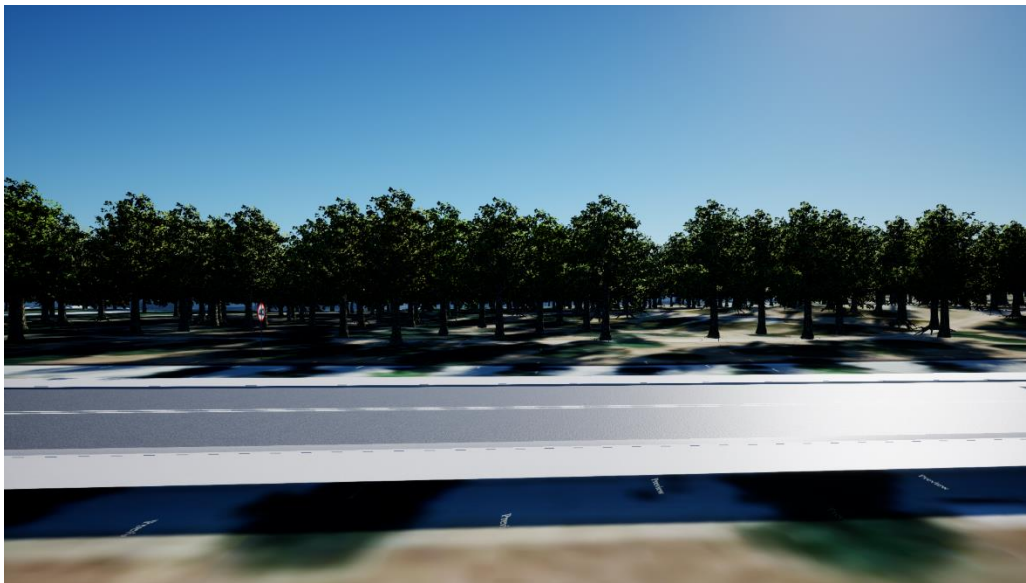


Figura 40. Prueba de cámara del dron

Como se ha visto en las imágenes, los resultados de la misión autónoma son satisfactorios, pudiéndonos mover a cualquier punto del sistema, aunque se ha utilizado este porque es sencillo obtener una distancia estimada de la carretera, y grabar en estos puntos del mismo.

7.2 Pruebas del seguimiento de un camino por los vehículos

En otro caso de prueba, se debe comprobar si los coches siguen de manera correcta el camino preestablecido en el sistema y si mantienen la velocidad que se establece como límite.

Para la prueba inicial de este seguimiento, se probó en un escenario base de Unreal Engine, proponiendo el siguiente camino a seguir:

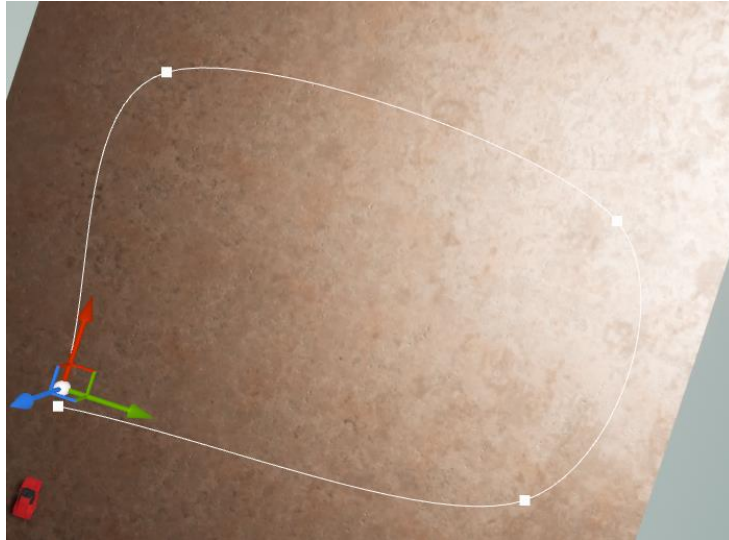


Figura 41. Camino de prueba para el seguimiento de los coches

Y, ejecutando la prueba se comprueba el seguimiento de este camino propuesto:



Figura 42. Imágenes del seguimiento de los vehículos.

Una vez comprobado este seguimiento del camino establecido para los vehículos, es necesario comprobar si se mantiene dentro de la velocidad que se ha establecido como límite y para ello se usará la información obtenida a través de la escritura del CSV.

En nuestra prueba, la velocidad límite establecida es de 45 y como vemos en la Figura 43 se mantiene más o menos en esa velocidad.

Y, de nuevo, como se ve en la Figura 43, también podemos comprobar que escribe toda la información necesaria, entre la que se encuentra la posición X, Y y Z del coche respecto a la del dron y por eso también aparece escrita la posición del dron, también se encuentra una marca temporal de cuando se escribió en el CSV, qué coche es, que hace uso de las etiquetas que se pueden dar a cada actor y un despliegue de la velocidad en cada eje.

1	X	Y	Z	Timestamp	Tag	Drone Position	X Velocity	Y Velocity	Z Velocity	XY Velocity
2	7852.314941	14533.3877	-1360.513062	25 ago. 2022 17:21:21:443	car1	X=7009.374 Y=10011.172 Z=1206.012	-0.227893	-1.912151	-0.008708	1.925703
3	7847.287598	14478.52344	-1361.097412	25 ago. 2022 17:21:22:406	car1	X=7010.749 Y=10012.359 Z=1206.822	-0.686308	-7.329649	-0.105313	7.362463
4	7840.42627	14372.86133	-1362.708496	25 ago. 2022 17:21:23:428	car1	X=7009.865 Y=10012.386 Z=1207.032	-0.647577	-11.367029	-0.233073	11.387846
5	7836.087402	14253.1084	-1364.634888	25 ago. 2022 17:21:24:457	car1	X=7007.458 Y=10010.724 Z=1207.442	-0.477491	-14.792787	-0.264548	14.802855
6	7833.909668	14089.79883	-1366.64563	25 ago. 2022 17:21:25:448	car1	X=7006.298 Y=10008.791 Z=1207.193	-0.154687	-19.098598	-0.179411	19.100069
7	7834.049316	13891.16309	-1369.653931	25 ago. 2022 17:21:26:477	car1	X=7006.781 Y=10008.722 Z=1205.889	0.054234	-23.211756	-0.355559	23.21454
8	7833.958496	13691.43457	-1372.717407	25 ago. 2022 17:21:27:420	car1	X=7008.059 Y=10010.143 Z=1203.811	-0.08684	-25.090687	-0.363128	25.093464
9	7830.121094	13435.25488	-1376.256348	25 ago. 2022 17:21:28:563	car1	X=7009.099 Y=10012.811 Z=1202.830	-0.643128	-28.903259	-0.398883	28.913164
10	7819.927734	13170.56445	-1379.827637	25 ago. 2022 17:21:29:561	car1	X=7009.015 Y=10013.221 Z=1202.881	-1.529351	-32.005222	-0.427386	32.04459
11	7802.388184	12877.85352	-1383.844727	25 ago. 2022 17:21:30:513	car1	X=7008.302 Y=10011.620 Z=1204.014	-2.161577	-34.014679	-0.358771	34.085182
12	7776.769043	12566.81641	-1388.220337	25 ago. 2022 17:21:31:453	car1	X=7007.430 Y=10009.898 Z=1205.375	-3.152537	-37.331715	-0.499104	37.467911
13	7743.047852	12227.0791	-1392.359253	25 ago. 2022 17:21:32:411	car1	X=7006.518 Y=10008.899 Z=1205.780	-4.411424	-40.316334	-0.431148	40.559258
14	7692.085449	11804.02637	-1396.700684	25 ago. 2022 17:21:33:473	car1	X=7006.559 Y=10009.270 Z=1205.446	-5.332089	-42.341572	-0.486958	42.678764
15	7649.161621	11430.10254	-1400.753662	25 ago. 2022 17:21:34:433	car1	X=7008.128 Y=10010.441 Z=1203.480	-4.832388	-45.182316	-0.475305	45.442482
16	7609.390625	11001.28711	-1405.272461	25 ago. 2022 17:21:35:446	car1	X=7010.439 Y=10012.103 Z=1203.230	-3.84543	-45.226044	-0.474068	45.391705
17	7575.942871	10583.88281	-1409.437134	25 ago. 2022 17:21:36:494	car1	X=7010.831 Y=10012.693 Z=1202.737	-3.471281	-44.413208	-0.402964	44.55048
18	7545.865234	10203.99609	-1412.734375	25 ago. 2022 17:21:37:433	car1	X=7009.679 Y=10012.136 Z=1200.977	-3.654157	-45.529545	-0.409694	45.677784
19	7510.584473	9784.188477	-1416.415283	25 ago. 2022 17:21:38:440	car1	X=7007.666 Y=10010.583 Z=1200.388	-3.849144	-44.390186	-0.363873	44.558243
20	7475.839844	9397.005859	-1419.390381	25 ago. 2022 17:21:39:411	car1	X=7006.521 Y=10008.991 Z=1199.645	-4.17397	-45.588421	-0.349515	45.780437
21	7436.352539	8976.257812	-1422.64209	25 ago. 2022 17:21:40:549	car1	X=7007.009 Y=10008.652 Z=1199.714	-4.199665	-44.06768	-0.324126	44.268528
22	7398.312988	8581.632812	-1425.290283	25 ago. 2022 17:21:41:490	car1	X=7008.322 Y=10010.067 Z=1200.583	-4.415077	-45.51281	-0.303395	45.727459
23	7361.556641	8200.506836	-1427.285034	25 ago. 2022 17:21:42:435	car1	X=7009.449 Y=10011.898 Z=1202.864	-4.360747	-44.196903	0.239805	44.412159
24	7316.824707	7752.926758	-1421.486084	25 ago. 2022 17:21:43:495	car1	X=7009.395 Y=10012.397 Z=1202.958	-4.340699	-45.547741	0.488302	45.756714
25	7281.249023	7368.353027	-1419.512085	25 ago. 2022 17:21:44:425	car1	X=7008.526 Y=10011.755 Z=1202.823	-4.418005	-44.597973	0.130151	44.81646
26	7235.694336	6954.609375	-1418.218262	25 ago. 2022 17:21:45:426	car1	X=7007.069 Y=10010.267 Z=1202.065	-5.001268	-44.570103	1.109424	44.863544
27	7185.574707	6533.594727	-1430.722412	25 ago. 2022 17:21:46:477	car1	X=7006.755 Y=10009.275 Z=1200.226	-6.277429	-44.515377	-1.787165	44.991322
28	7116.76416	6117.060059	-1438.72876	25 ago. 2022 17:21:47:459	car1	X=7007.448 Y=10009.562 Z=1200.023	-8.107555	-44.49551	-0.3213	45.229259

Figura 43. Información que proporciona el CSV

7.3 Pruebas del algoritmo de estimación de velocidad

Para terminar con las pruebas, hay que comprobar que funciona tanto la detección de los vehículos como el tracking y realizar una comparación de los resultados obtenidos con los reales escritos en el CSV del vehículo.

Como ya se ha comentado, para obtener una estimación de la velocidad a la que va el vehículo se tiene que calcular la distancia de manera estimada utilizando elementos del entorno que son visibles en la cámara del dron. Por ello, para todas las pruebas se ha utilizado el mismo tramo de carretera y por tanto la misma distancia estimada para así poder comparar el análisis de velocidad y del algoritmo en sí en las mismas condiciones para todas las pruebas.

La distancia estimada utilizada para todas las pruebas realizadas ha sido la siguiente:



Figura 44. Distancia estimada utilizada

Es decir, 23,36 metros, esta distancia ha sido estimada mediante elementos de la simulación como un banco en el extremo derecho del vídeo, que es donde está el final del tramo, una señal de tráfico y la propia sombra de los árboles de la esquina inferior izquierda. Como se ha utilizado la mitad del vídeo para poder calcular velocidades de vehículos en ambas direcciones y por tanto con una simple regla de tres, la distancia resultante estimada final es de 13,04 metros, no es la mitad exacta puesto que si bien el banco se encuentra a la derecha de la cámara no es exactamente el final y se ha calculado mediante la ubicación dentro del vídeo estando el banco en el punto X 1720 y la mitad siendo 960, calculando así la distancia.

Antes de comenzar el análisis de resultados es necesario comentar que los datos de tiempo son los segundos del tiempo en el que fue probado el sistema.

7.3.1 Prueba de detección en un entorno real

Para comprobar que el software funcionaba bien en primera instancia, se hizo una prueba con un vídeo de una carretera real [55] y comprobando que la detección de los vehículos y que su tracking fuese correcto. Entonces, utilizando el método comentado anteriormente, aunque sin poder obtener la velocidad puesto que es una carretera desconocida y no se poseen datos sobre ella.

En la siguiente imagen se puede comprobar que la detección se realiza correctamente:

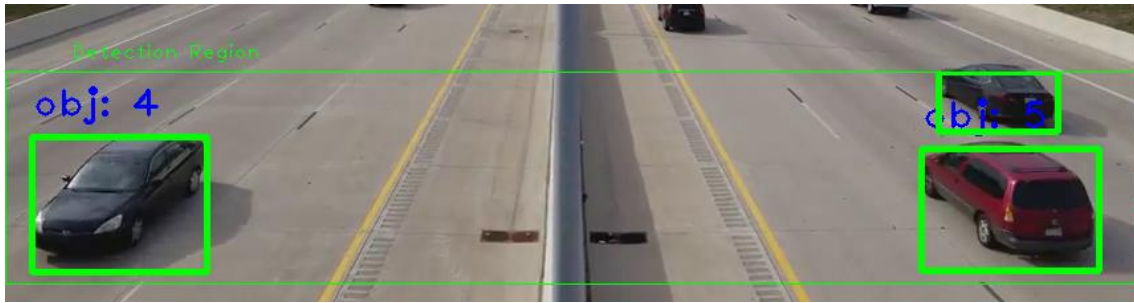


Figura 45. Prueba en un entorno real.

Una vez comprobado su funcionamiento en un entorno real y viendo que hace una detección correcta en la zona deseada se ha pasado a probar el software en el entorno simulado.

7.3.2 Prueba con un solo vehículo

Para la prueba de un solo vehículo, se ha colocado un coche en la parte izquierda que sigue un camino hasta la derecha de la carretera pasando por el campo de visión del dron.

Sacando como resultado lo siguiente:

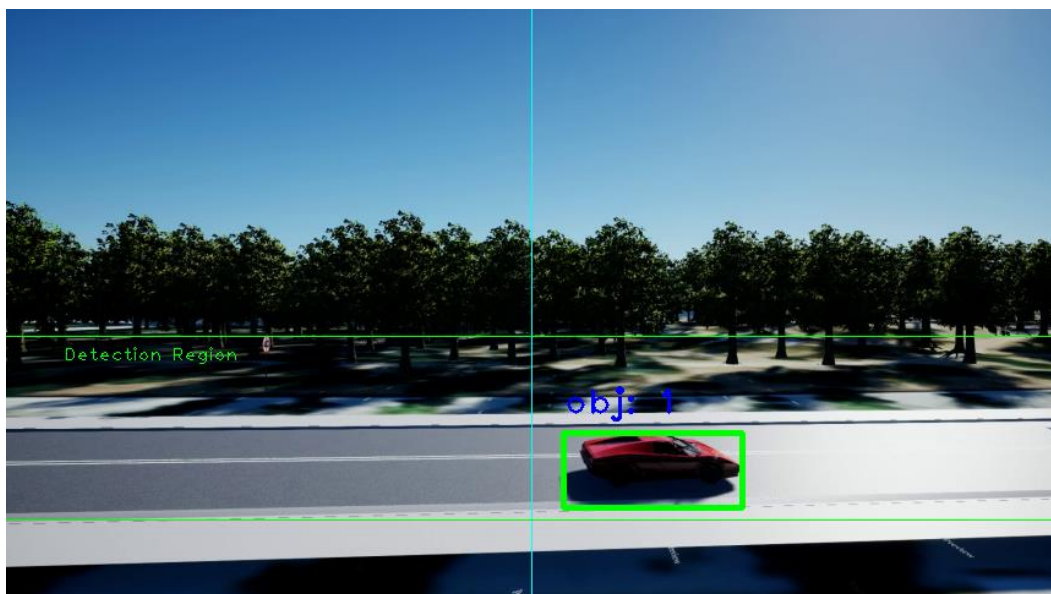


Figura 46. Prueba con un solo vehículo

Como se ve en la anterior Figura, la detección la realiza de manera correcta y mantiene el ID 1 a través de todo el vídeo. Los resultados de tiempo son los siguientes:

ID	Primera detección del vehículo	Primera detección del vehículo tras sobrepasar la distancia estimada	Diferencia de tiempo (segundos)
1	34.264733	35.231700	0,967

Tabla 14. Marcas de tiempo prueba de un vehículo.

Por lo tanto, el tiempo que tarda en recorrer en el vídeo esa distancia es de 0,967 segundos y utilizando los 13,04 metros estimados anteriormente se obtiene una velocidad de 13,49 metros/segundo y convirtiéndolo a kilómetros/hora se obtiene 48,55 km/h.

7.3.3 Prueba con dos vehículos

En esta prueba se han colocado dos coches en ambas direcciones de la carretera de manera simultánea y ha dado como resultado lo que se ve en la siguiente imagen:



Figura 47. Prueba con dos vehículos

Sus IDs son tan altas porque al tener ya más objetos en pantalla, a la hora de iniciar la ejecución hay partes de la carretera que toma de manera muy breve como objeto y esto hace aumentar el número del ID, haciendo que cuando se detecte el vehículo de manera completa sea momentáneamente posterior y por tanto los resultados sean algo peores que con un solo vehículo, aunque sigan siendo más o menos satisfactorios.

ID	Primera detección del vehículo	Primera detección del vehículo tras sobrepasar la distancia estimada	Diferencia de tiempo (segundos)
6	48,111706	48.942706	0,831
7	47.248695	48.213249	0,965

Tabla 15. Marcas de tiempo prueba de dos vehículos.

Por lo tanto, el primer vehículo con ID 7 que va de derecha a izquierda tarda un tiempo de 0,965 segundos y entonces, va a una velocidad de 13,52 metros/segundo, que convirtiéndolos son 48,67 km/h.

Y, el vehículo con ID 6 que va de izquierda a derecha tarda un tiempo de 0,831 y su velocidad es de 15,69 que convirtiéndolo a kilómetros/hora es de 56,49 km/h.

7.3.4 Prueba con cuatro vehículos

Para esta prueba se han colocado dos vehículos en cada carril de la carretera moviéndose en direcciones contrarias para que aparezcan todos de manera simultánea dentro de la grabación del dron.

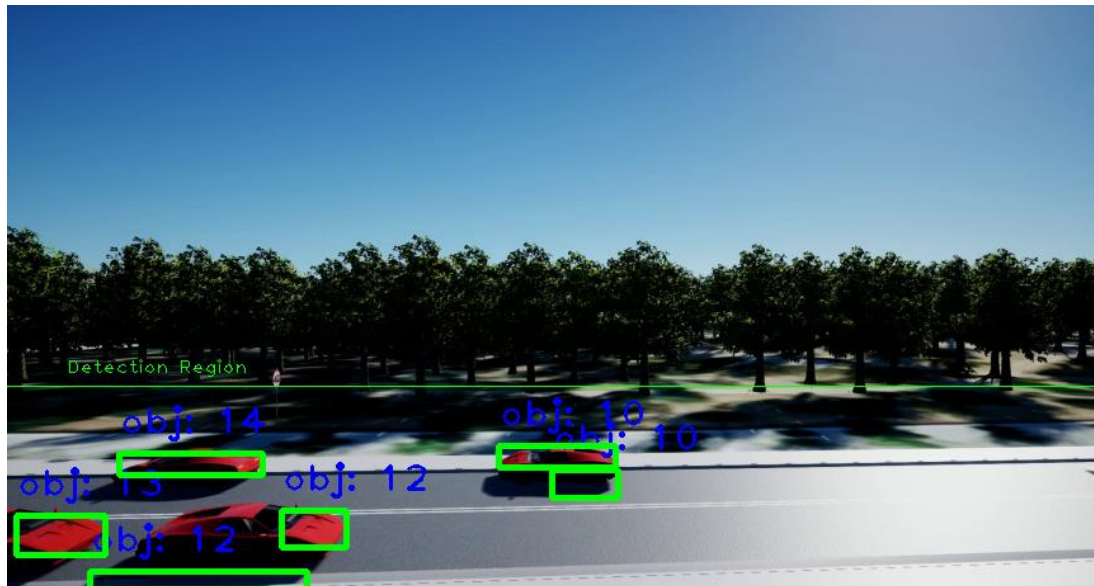


Figura 48. Prueba con 4 vehículos.

De igual manera que el anterior, al haber más objetos, existe una mayor interferencia con el sustractor y genera más ruido en la imagen generando IDs en partes de la imagen que no son necesariamente objetos. La opción de detectar las sombras está activada y por eso en algunos de ellos los toma como el mismo objeto, pero con bounding box distinta.

ID	Primera detección del vehículo	Primera detección del vehículo tras sobrepasar la distancia estimada	Diferencia de tiempo (segundos)
10	25.612240	26.83886	1,22
12	26.456294	27.295897	0,84
13	26.711911	27.527280	0,815
14	25.365089	26.368698	1,003

Tabla 16. Marcas de tiempo prueba de cuatro vehículos.

Por lo tanto, a través de estos datos, se puede conseguir lo siguiente:

El primer vehículo que va de derecha a izquierda, que tiene un ID de 14, tarda un total de 1,003 segundos en recorrer el tramo y sabiendo que es de 13,04 metros, va a una velocidad de 12,993 metros/segundo, que convirtiéndolo se queda en 46,78 km/h.

El segundo vehículo del carril del fondo que se desplaza de derecha a izquierda, con un ID de 10, tarda 1,22 segundos y por tanto va a una velocidad de 10,63 metros/segundo, es decir de 38,27 km/h.

El tercer vehículo en la imagen, que va de izquierda a derecha con un ID de 12, tarda 0,84 segundos en recorrer el tramo y va a una velocidad de 15,53 metros/segundo y por tanto es de 55,91 km/h.

El cuarto vehículo, también de izquierda a derecha, con un ID de 13, tarda 0,815 segundos en recorrer el tramo específico y por tanto se desplaza a una velocidad de 15,99 metros/segundo y de 57,57 km/h.

7.3.5 Prueba desde encima de la carretera

Para el último conjunto de pruebas se ha colocado el dron desde encima de la carretera y mediante el video resultante se ha procesado el vídeo y como se observa en la siguiente imagen, la detección sigue siendo correcta:



Figura 49. Prueba con la cámara encima de la carretera

Una vez procesado el vídeo, los resultados de este son aproximadamente los mismos que en las pruebas anteriores por eso no se va a realizar un análisis de tiempos y velocidad, al menos de manera escrita, porque sería redundante, pero hay que mencionar que esos datos resultantes son parecidos a los anteriores.

7.3.6 Evaluación final

Para acabar con el apartado de evaluación del algoritmo, se va a hacer una comparación de las velocidades medias reales con las velocidades medias estimadas en cada caso, resaltando el margen de error existente en el algoritmo y que ha ocurrido con el ruido en los videos.

Por lo tanto, el siguiente gráfico muestra una comparación entre las dos velocidades comentadas:

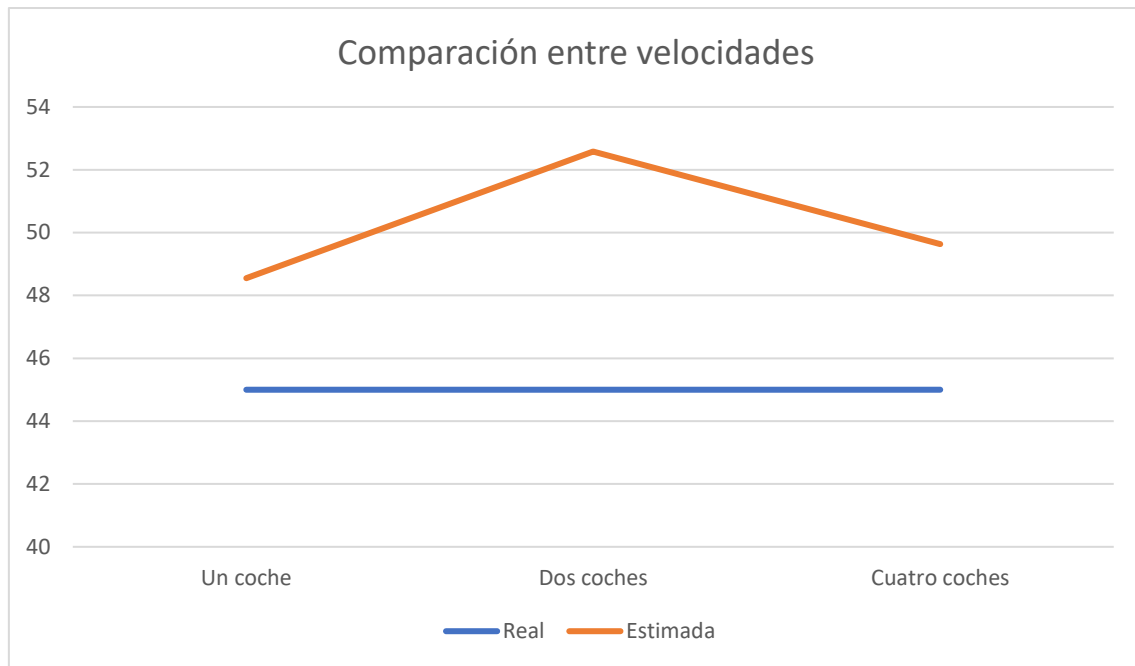


Figura 50. Gráfico de comparación entre velocidades.

De este gráfico se puede obtener que cada uno de los casos de pruebas tienen los siguientes márgenes de error en comparación con la velocidad real que se ha propuesto como límite, que en todos los casos ha sido de 45 km/h, el cual se sabe porque se ha revisado en los datos mostrados en el CSV y porque el sistema está diseñado para que los vehículos mantengan una velocidad fija:

Un coche: 7,89%

Dos coches: 16,85%

Cuatro coches: 10,3%

Lo que se obtiene de esto es que no es un algoritmo exacto, pero se obtienen resultados lo suficientemente fidedignos al dato original para considerarlo válido. Pero hay que aclarar que, si bien puede parecer que aumentar los coches no empeora los resultados debido a que tiene un margen de error mejor que el de dos coches, no es verdad puesto que se ha observado que a mayor cantidad de coches hay más cantidad de ruido en cada frame y puede dar como resultado errores en el tracking.

Otro aspecto que se ha observado es que si los vehículos están muy juntos puede llegar a considerarlo como un único objeto detectado y por eso es mejor dejar algo de separación entre ellos en las pruebas. Además, se ha reconocido que los coches del carril más cercano a la cámara, es decir el carril que va de izquierda a derecha desde la cámara horizontal, tienen resultados peores, en las pruebas con el dron encima de la carretera ocurre lo mismo y es por el ruido que debe generar el terreno del simulador cercano a ese carril.

8 CONCLUSIONES Y TRABAJOS FUTUROS

Como conclusión a todo el proyecto se va a hacer un pequeño resumen de lo que se ha cumplido en el mismo de lo que se había propuesto y se hará una pequeña conclusión personal del autor, dejando claro que le ha parecido todo el proceso, cosas que cambiaría o que no. Para terminar con este apartado, y con el documento en sí, se comentará algunos caminos que se pueden seguir para avanzar con lo propuesto en este proyecto para mejorarlo en posibles trabajos futuros.

8.1 Conclusiones del proyecto

En cuanto a las conclusiones que podemos obtener de todo lo desarrollado en el proyecto, podemos deducir que se ha obtenido conocimientos previos tanto de algoritmos por visión por computador como del funcionamiento general de Unreal Engine y de los drones como vehículo no tripulado, incluyendo todos sus aspectos.

Como bien se ha comentado para la realización del documento se han cumplido todos los objetivos propuestos y se ha conseguido realizar un explicativo del proyecto que detalla todo lo relacionado al mismo.

Pero por otro lado, no todo es la escritura del documento y por eso en cuanto a los objetivos propuestos para el apartado de software se ha cumplido casi todo lo propuesto inicialmente, como bien se ha comentado se intentó la detección de las líneas de la carretera pero no se consiguió un resultado lo suficientemente bueno para poder incluirlo, pero en cuanto a los demás objetivos han sido cumplido todos teniendo un sistema que posee vehículos que pueden seguir camino a una velocidad determinada y que posteriormente las grabaciones que se han hecho mediante la cámara del dron son procesadas y se pueden detectar estos vehículos y estimar su velocidad.

Referido a los resultados obtenidos en las pruebas, como ya se mostrado anteriormente en el apartado de evaluación, son lo suficientemente satisfactorios para considerarlo válido, pero el ruido que generan las imágenes haciendo que los IDs no sean únicamente los correspondientes a los coches hace pensar que el sistema no sea escalable a una carretera muy concurrida.

Por lo tanto, para terminar con este apartado, el sistema resultante es válido y funcional, y a partir de él se ha podido explicar y mostrar unos conceptos que han servido tanto al autor de este, a la hora de entender mejor el desarrollo y poder hacerlo lo mejor posible, como al posible lector para que su lectura sea lo suficientemente entendible aún sin saber nada de ningún campo relacionado a los presentes en el proyecto.

8.2 Conclusiones personales

Este apartado va a ser escrito desde una vista en primera persona, pues al fin y al cabo es la opinión del autor.

Me gustaría empezar poniendo las sensaciones que he tenido durante el desarrollo del trabajo. Para empezar, creo que con sus más y sus menos el proyecto ha sido disfrutable, ha habido partes más agradables de realizar y que me han resultado más satisfactorias y otras que menos, pero viéndolo como un todo creo que lo he acabado disfrutando.

También, hay que mencionar que yo nunca había tocado nada relacionado con Inteligencia Artificial a este punto, solo tenía conocimiento de algoritmos como el A estrella y demás pero muy poco en general y creo que si hubiese tenido algo más de base podría haberme frustrado algo menos durante la realización de este, porque si, al final hubo un punto en el que me frustré en su desarrollo, después de muchos intentos probando algoritmos y demás para tener una detección óptima, pero una vez pasado este punto todo fue algo más ameno.

Si bien la parte de Inteligencia Artificial se me atragantó un poco más, la parte de utilizar el Unreal Engine programando con las 'blueprints' y preparando el entorno simulado ha sido, sin duda, la que más he disfrutado porque a mí siempre me han gustado los videojuegos y el echar un pequeño vistazo a los nodos que utiliza Unreal para la programación de estos me ha gustado bastante.

Otro punto a tener en cuenta es que según como están hecho la programación base de la universidad con respecto a las prácticas y a las asignaturas del último cuatrimestre, es muy poco factible conseguir sacar el tiempo suficiente para tenerla acabada para la convocatoria de junio, a no ser que consigas prácticas en el verano anterior, se me hizo muy complicado compaginar todo y, por eso, se ha tenido que retrasar a septiembre. Por eso creo que esto es un punto que se podría mejorar.

Para acabar, quiero decir que, si bien ha habido complicaciones en el desarrollo, he disfrutado el camino para completar el trabajo y que he aprendido bastante de áreas y conceptos que son muy interesantes.

8.3 Trabajos futuros

Para acabar con este capítulo, se va a hacer un breve paso por los posibles caminos futuros a seguir, pues no es una explicación de estos si no mostrarlos y dejar constancia que están ahí.

El primer camino que se podría seguir es el de conseguir que YOLO funcionase dentro del entorno simulado, el cómo eso ya no lo tengo muy claro porque no se ha conseguido en este proyecto, pero es un apartado que podría mejorar bastante la detección si se consiguiese, pues es un software muy útil.

Otro posible camino a seguir sería el tener información de la carretera de antemano utilizando la tecnología XODR, que nos permite si tuviésemos el mapa en formato xodr, mapear las carreteras existentes en el mapa y obtener datos que se hayan introducido en el mapa como velocidad en ese tramo y demás, es una buena manera de tener más información de la carretera, pero debido a cuando se tuvo constancia de esta tecnología no dio tiempo a ahondar en ella.

Y, por último, se podría investigar como detectar las líneas de la carretera, tratando de colocar la cámara desde donde puedan verse y haciendo el simulador lo más realista posible para que así la detección de estas líneas sea posible y de este modo conseguir detectar una infracción más.

Quitando la parte de XODR, lo demás son cosas que han sido probado dentro de este proyecto, pero por las circunstancias del mismo, ya sea por el simulador o el ruido que generan las imágenes del dron u otros aspectos han hecho que no se haya conseguido realizarlo y por eso está bien señalarlo como posibles trabajos futuros.


9 BIBLIOGRAFÍA

- [1] «Los drones de Google llegan a las 200.000 entregas comerciales», 1 de marzo de 2022. <https://www.whatsnews.com/2022/03/01/los-drones-de-google-llegan-a-las-200-000-entregas-comerciales/> (accedido 18 de agosto de 2022).
- [2] «Importancia de los drones en la actualidad y qué podemos esperar de ellos en el futuro», *Dronexperts*, 8 de abril de 2021. <https://dronexperts.es/importancia-de-los-drones-en-la-actualidad-y-que-podemos-esperar-de-ellos-en-el-futuro/> (accedido 18 de agosto de 2022).
- [3] J. Martín, «Drones y radares de la DGT: la última amenaza para el conductor», *Motorpasión*, 8 de agosto de 2021. <https://www.motorpasion.com/revision/drones-radares-dgt-ultima-amenaza-para-conductor> (accedido 18 de agosto de 2022).
- [4] Equipo de Expertos de Ciencia y Tecnología de la Universidad Internacional de Valencia., «¿Qué es un dron y cómo funciona?», *Nuestros expertos*, 21 de marzo de 2018. <https://www.universidadviu.com/int/actualidad/nuestros-expertos/que-es-un-dron-y-como-functiona>
- [5] «Title: How drones can improve winemaking - Terraviva». <https://www.google.com/imgres> (accedido 15 de julio de 2022).
- [6] «UAV Navigation in Depth: What is an IMU and what is it used for? | UAV Navigation». <https://www.uavnavigation.com/company/blog/what-is-an-IMU> (accedido 6 de agosto de 2022).
- [7] Roberto, «¿Qué tipos de vuelo puedo realizar con mi dron?», *Gis&Beers*, 24 de noviembre de 2019. <http://www.gisandbeers.com/tipos-de-vuelo-puedo-realizar-dron/> (accedido 6 de agosto de 2022).
- [8] *Airplane control - Roll, Pitch, Yaw*, (25 de marzo de 2015). Accedido: 6 de agosto de 2022. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=pQ24NtnaLl8>
- [9] A. Rodríguez, «Tipos de drones y sus características | Iberfdrone Formación de pilotos», *Iberfdrone*, 3 de octubre de 2019. <https://iberfdrone.es/tipos-drones-y-caracteristicas/> (accedido 2 de agosto de 2022).
- [10] «Qué son y cómo funcionan los enjambres de drones», *AERTEC*, 7 de febrero de 2016. <https://aertecolutions.com/2016/02/08/enjambre-de-drones/> (accedido 2 de agosto de 2022).
- [11] «Entender los sensores para drones», *Drones, Cámaras, Acción*, 20 de octubre de 2020. <https://drones-cameras-accion.com/blog/entender-los-sensores-para-drones/> (accedido 3 de agosto de 2022).
- [12] webmaster, «Para qué sirven los GPS en los drones y en qué situaciones se utilizan», *¿Quieres ser piloto de drones?*, 23 de octubre de 2021.

<https://cursodedrones.es/para-que-sirven-los-gps-en-los-drones-y-en-que-situaciones-se-utilizan/> (accedido 4 de agosto de 2022).

[13] «LiDAR: qué es y qué utilidad tiene para la fotogrametría», *Pix4D*. <https://www.pix4d.com/es/blog/lidar-fotogrametria> (accedido 3 de agosto de 2022).

[14] «▷ Drone con Cámara Térmica | El Mejor Dron Termográfico 2022», *Cámara Térmica*. <https://www.camaratermica.info/drone-con-camara-termica/> (accedido 3 de agosto de 2022).

[15] P. Migueldronero, «Controladora de Vuelo (FC) ¿Que es? », *De Cero a Dronero*, 30 de septiembre de 2020. <https://deceroadronero.es/componentes-que-tienes-que-saber-de-tu-dron/controladora-de-vuelo-fc-que-es/> (accedido 3 de agosto de 2022).

[16] Dronecode, «PX4». Dronecode, Linux Foundation. [En línea]. Disponible en: <https://px4.io/software/software-overview/>

[17] «Introducción al Controlador de Vuelo de Drones PIXHAWK - Hardware libre», *gidahatari*. <https://gidahatari.com/ih-es/introduccion-al-controlador-de-vuelo-de-drones-pixhawk-hardware-libre> (accedido 15 de julio de 2022).

[18] D. Technologies, «Why We Chose PX4 (vs APM) as Luci's Default Firmware», *Medium*, 8 de agosto de 2016. <https://medium.com/@Dronesmith/why-we-chose-px4-vs-apm-as-lucis-default-firmware-ea39f4514bef> (accedido 3 de agosto de 2022).

[19] Lorenz Meier, «Guía de desarrollador de MAVLink». [En línea]. Disponible en: <http://mavlink.io/en/>

[20] Dronecode, «Librería MavSDK». Dronecode, Linux Foundation. [En línea]. Disponible en: <https://mavsdk.mavlink.io/main/en/index.html>

[21] Microsoft, «AirSim». [En línea]. Disponible en: <https://microsoft.github.io/AirSim/?msclkid=1b5e9020c0bb11ec8b769aed508706eb>

[22] *AirSim Demo*, (15 de febrero de 2017). Accedido: 15 de julio de 2022. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=-WfTr1-OBGQ>

[23] Microsoft, «Uso de PX4 en AirSim». [En línea]. Disponible en: https://microsoft.github.io/AirSim/flight_controller/

[24] «Aerial Autonomy: Project AirSim», *Microsoft AI*. <https://www.microsoft.com/en-us/AI/autonomous-systems-project-airsim> (accedido 4 de agosto de 2022).

[25] «Features -- Gazebo». <https://gazebo.org/features> (accedido 4 de agosto de 2022).

[26] «[ROS Q&A] 191 - How to Launch the Parrot Drone Simulation Locally», *The Construct*, 30 de agosto de 2019. <https://www.theconstructsim.com/how-to-launch-drone-simulation-locally/> (accedido 4 de agosto de 2022).

[27] Epic Games, «Unreal Engine». Epic Games. [En línea]. Disponible en: <https://www.unrealengine.com/en-US>

- [28] «Cesium». [En línea]. Disponible en: <https://www.cesium.com/>
- [29] «Cesium for Unreal», *Cesium*. <https://cesium.com/platform/cesium-for-unreal/> (accedido 15 de julio de 2022).
- [30] Intel, «OpenCV». OpenCV.org. [En línea]. Disponible en: <https://www.opencv.ai/>
- [31] C. Hillary, «Getting Started on Object Detection with openCV», *Analytics Vidhya*, 22 de diciembre de 2019. <https://medium.com/analytics-vidhya/getting-started-on-object-detection-with-opencv-5962a75876a6> (accedido 19 de julio de 2022).
- [32] «¿Qué es la inteligencia artificial (IA)?» <https://www.oracle.com/es/artificial-intelligence/what-is-ai/> (accedido 19 de julio de 2022).
- [33] IBM, «¿Qué es Computer Vision?» <https://www.ibm.com/topics/computer-vision#:~:text=%20Here%20are%20a%20few%20examples%20of%20established,an%20object%20once%20it%20is%20detected.%20More%20?msclkid=43f49fbfc0c211ecbd9dfaf3aff099d8>
- [34] IBM, «¿Qué es Deep Learning?» <https://www.ibm.com/mx-es/cloud/deep-learning>
- [35] R. KeepCoding, «¿Qué son las Redes Neuronales Convolucionales? | KeepCoding Tech School», 11 de noviembre de 2020. <https://keepcoding.io/blog/redes-neuronales-convolucionales/> (accedido 19 de julio de 2022).
- [36] *Cars Detection Output using // opencv // Python /// for my github*, (9 de mayo de 2020). Accedido: 15 de julio de 2022. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=R5txKotYeMM>
- [37] H. Parmar, «Detect speed of a car with OpenCV in Python», *CodeSpeedy*, 16 de abril de 2020. <https://www.codespeedy.com/detect-speed-of-a-car-with-opencv-in-python/> (accedido 19 de julio de 2022).
- [38] «OpenCV Thresholding (cv2.threshold)», *PyImageSearch*, 28 de abril de 2021. <https://www.pyimagesearch.com/2021/04/28/opencv-thresholding-cv2-threshold/> (accedido 19 de agosto de 2022).
- [39] T. rédac, «Descubra el algoritmo KNN: un algoritmo de aprendizaje supervisado», *Formation Data Science | DataScientest.com*, 28 de diciembre de 2021. <https://datascientest.com/es/que-es-el-algoritmo-knn> (accedido 19 de agosto de 2022).
- [40] «Background Subtraction - an overview | ScienceDirect Topics». <https://www.sciencedirect.com/topics/engineering/background-subtraction> (accedido 19 de agosto de 2022).
- [41] E. Laine, «Answer to “What are createBackgroundSubtractorKNN parameters in OpenCV C++?”», *Stack Overflow*, 26 de abril de 2017. <https://stackoverflow.com/a/43636256> (accedido 19 de agosto de 2022).

- [42] X. Liu y Z. Zhang, «A Vision-Based Target Detection, Tracking, and Positioning Algorithm for Unmanned Aerial Vehicle», *Wirel. Commun. Mob. Comput.*, vol. 2021, pp. 1-12, abr. 2021, doi: 10.1155/2021/5565589.
- [43] X. Hou, Y. Wang, y L.-P. Chau, «Vehicle Tracking Using Deep SORT with Low Confidence Track Filtering», en *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, sep. 2019, pp. 1-6. doi: 10.1109/AVSS.2019.8909903.
- [44] «The UA-DETRAC Benchmark Suite». <https://detrac-db.rit.albany.edu/> (accedido 27 de agosto de 2022).
- [45] «BOE-A-2021-4477.pdf». Accedido: 20 de agosto de 2022. [En línea]. Disponible en: <https://www.boe.es/boe/dias/2021/03/22/pdfs/BOE-A-2021-4477.pdf>
- [46] Gobierno de España, *Impuesto sobre el Valor Añadido*. [En línea]. Disponible en: https://www.boe.es/biblioteca_juridica/codigos/codigo.php?id=057_Impuesto_sobre_el_Valor_Anadido&modo=2
- [47] Dron Europa, «Actividades no EASA», *Dron Europa*. <https://www.droneuropa.com/operadora-drones/actividades-NO-EASA.php>
- [48] One Air, «NUEVO REGLAMENTO EUROPEO DE DRONES 2021», *One Air*. <https://www.oneair.es/nuevo-reglamento-europeo-drones/>
- [49] Gobierno de España, *Ley Orgánica de Protección de Datos*. [En línea]. Disponible en: <https://www.boe.es/eli/es/lo/2018/12/05/3/con>
- [50] S. S. A. E. de ingeniería industrial eléctrica e investigador de seguridad en el equipo de I. L. C. de Telefónica, «Menos accidentes de tráfico gracias a la Inteligencia Artificial», *Think Big*, 30 de marzo de 2021. <https://empresas.blogthinkbig.com/menos-accidentes-de-trafico-gracias-a-la-inteligencia-artificial/> (accedido 6 de agosto de 2022).
- [51] P. G. D. de Atauri, «¿Multas por Inteligencia Artificial? A propósito del procedimiento sancionador promovido por actuación administrativa automatizada», *El Foro de Labos*, 22 de septiembre de 2021. <https://www.elforodelabos.es/2021/09/multas-por-inteligencia-artificial-a-proposito-del-procedimiento-sancionador-promovido-por-actuacion-administrativa-automatizada/> (accedido 6 de agosto de 2022).
- [52] DGT, «Tráfico distribuye los 39 drones que vigilarán las carreteras.», 7 de julio de 2021. [En línea]. Disponible en: <https://revista.dgt.es/es/noticias/nacional/2021/07JULIO/0707Drones-vigilancia-carreteras.shtml#:~:text=Desde%20el%20inicio%20de%20la%20actividad%20de%20vigilancia,veh%C3%ADculos%20y%20ha%20detectado%20m%C3%A1s%20de%20600%20infracciones.>
- [53] Lisa Institute, «Drones: ventajas y usos potenciales para la Policía, Seguridad Privada, Emergencias y Bomberos», mar. 2019. [En línea]. Disponible en: <https://www.lisainstitute.com/blogs/blog/drones-usos-policia-seguridad-emergencias-bomberos>

[54] J. Díaz, «Fabricar un coche eléctrico contamina un 70% más que uno de gasolina», *elconfidencial.com*, 23 de noviembre de 2021. https://www.elconfidencial.com/tecnologia/novaceno/2021-11-23/coche-electrico-co2-contaminacion-gasolina_3329281/ (accedido 6 de agosto de 2022).

[55] *4K Video of Highway Traffic!*, (21 de octubre de 2017). Accedido: 25 de agosto de 2022. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=KBsqQez-O4w>

ANEXO A. RESUMEN EN INGLÉS

Before starting with this annex, it should be noted that what is going to be shown is a summary of the whole document and, as it is a summary, it is not going to be very extensive in all the sections, taking only the information that is essential for it.

A.1 INTRODUCTION

Drones provide information that humans are unable to see, they have sensors of different types that provide data that can be further processed to be used in different tasks, whether it be agriculture, with crop control, photography or cinema, with the possibility of taking camera shots that would otherwise be much more complicated or, as in the case presented here for police or rescue tasks, either with the thermal sensor to help in fires or with the camera itself to detect traffic offences of all kinds.

In this project we are going to focus on this last-mentioned field, this is because it is a field that is booming in Spain with the Dirección General de Tráfico allocating more and more resources to this type of method for detecting infractions.

With the aforementioned, one of the main motivations of the project can be commented on, which is that the current drones used for road control do not use radars to measure the speed of the vehicles that travel those stretches that are being monitored and therefore can only take into account other types of infractions such as not wearing a seat belt or talking on a mobile phone, although these detections are always subject to failure and therefore must be checked manually by a DGT technician.

Due to the impossibilities in Spain to fly in a large number of places, it is beneficial to consider the use of a simulated environment to be able to test with a margin of error in terms of flight and with a much larger number of tests, because if you had to go to a road every time you wanted to test something new of the algorithm it would be a costly job.

This simulation will be carried out with AirSim using the PX4 software as a flight controller, thus managing the drone hardware and processing the instructions it receives. Furthermore, it has not only been left with the simulation presented by AirSim, but to make it more realistic, Unreal Engine has been used with the Cesium extension to be able to use a map of the world, specifically in this work the Colmenarejo campus of the Carlos III University of Madrid.

To conclude this section, the final objective of the project is to be able to estimate speed by using computer vision and related algorithms. To do this, we have made use of all of the above, in addition to a study of different options and an analysis of both the state of the art and the design of the whole system.

A.2 STATE OF THE ART

In this chapter, various concepts that are useful for the implementation of this system and that must be known in order to understand it without causing loss in the text will be described.

A.2.1 DRONE

To begin with, we will describe what a drone is and why it is useful not only for our work but also for others.

A drone or also called in some contexts UAV is a remotely controlled vehicle that uses the rotational force of the engines attached to its propellers to rise and perform flights.

Drones emerged in the military field to carry out reconnaissance missions or to attack the enemy. But nowadays they are much more standardised in the population and are even used for leisure purposes. Although they are highly regulated so that they are not misused.

They have several flight profiles that are defined as the state of the drone, i.e. ascent, descent or hovering to give an example, but they also have several movements that allow the drone to move freely [7]:

- **Yaw:** This refers to the movement that a drone can perform on its vertical axis, being able to rotate 360 degrees from right to left or vice versa the direction in which it moves. This is a particularly important movement as it allows the direction of the drone to be changed and the front of the drone, where the camera is usually placed, to be oriented in the direction in which the movement is to be made.
- **Roll:** This movement allows the drone to be tilted to the right or left and allows it to glide by making slight turns to the right or left.
- **Pitch:** This is the analogous movement to the roll, but instead of turning from right to left, it turns from top to bottom. It allows upward and downward dive movements to change altitude.

It is also worth highlighting the numerous sensors that drones have that help to carry out tasks for them, among which the following stand out [11]:

- **GPS:** a sensor capable of determining the position in coordinates of the device using it. [12]
- **IMU:** It is capable of collecting data through different sensors that are integrated in it and by processing this data with the CPU it can estimate information such as altitude, position and speed. [6]
- **LiDAR:** It takes its name from its acronym in English Light Detection and Ranging, this technology makes use of fast laser pulses to measure the time it takes between releasing the pulse and when it is reflected on a surface or object

and the pulse is received again, through this time it is able to calculate the distance to the required object.

- **Thermal sensor:** sensors capable of measuring the temperature of surfaces and objects by picking up the heat signature emitted by the object.
- **Hyperspectral and multispectral sensor:** Multispectral sensors are a type of sensor capable of recognising not only the wavelength of the three basic colours perceived by humans (red, green, and blue) but also wavelengths that humans cannot perceive such as near infrared radiation and shortwave infrared radiation. While hyperspectral sensors are much more powerful versions of the former.

A.2.2 SIMULATION

As mentioned above, in order to avoid problems when testing the drone tasks, a simulated environment using AirSim has been created.

AirSim is a simulator for drones, cars and other vehicles built in Unreal Engine. It is very useful for projects such as the one being presented, as it is open-source, cross-platform software with the ability to perform software-in-the-loop simulations with flight controllers such as PX4 and ArduPilot and hardware-in-the-loop simulations with PX4. [21]

As already mentioned, PX4 is used as software that emulates a flight controller, which is an integrated card inside the drone that contains a processor that is able to take the orders that come to the drone through the receiver and process them to perform the action correctly and PX4 is useful for its ease of use within Airsim and for the management of the drone and its tasks. Therefore, it has been concluded that PX4 was the right software for this project. For the drone to receive these commands and process them, MavSDK and MAVLink are also used. [16, p. 4]

Starting with MavLINK, this is a very lightweight messaging protocol for communication with drones and between its components that follows a modern design pattern that mixes the publish and subscribe pattern and the point-to-point pattern. The data stream is sent and published as a topic while the sub-protocols are sent point-to-point. [19]

MavSDK is a collection of libraries for various programming languages to interface to systems using MAVLink. It features functionality such as the ability to move the drone or elevate it, use the drone's telemetry and control the information it handles. [20]

AirSim is built in Unreal Engine, and the great thing about Unreal is that due to its high level of realism, simulations of this type are easier than in other engines, and there is currently a free version of it, making it easier to use for smaller or academic projects. It also contains a high degree of portability, with programs created from it being able to be used on multiple platforms such as Windows, Linux, PlayStation. [27]

In addition, for the most realistic simulation possible, the Cesium extension has been used, which allows exact copies of cities around the world to be made if you had the

computing power to render them all. It is accurate, with great performance when using large amounts of 3D objects such as buildings or roads. [29]

A.2.3 COMPUTER VISION

Bearing in mind that the final objective of the system is the detection and tracking of vehicles in order to subsequently be able to estimate their speed by processing the frames of a video, it is necessary to explain a series of related concepts in order to understand them.

The detection of objects in a video consists of the use of algorithms to be able to recognise objects of all types, in this case vehicles. Normally, masks are used, which are mechanisms to indicate which pixels are to be maintained when applying it and which are not in order to make a better distinction in terms of the objects of interest, which are applied to each of the frames of a video.

In terms of tracking, a series of algorithms are applied to follow an object through the frames of the video so that this object that has been detected is not considered a new object in each new frame. To do this, identifiers are normally assigned to each object and, if the algorithm is well performed, it will continue to track the object throughout the video by 'tracking' it through the centroid of the object's bounding box, which is basically a rectangle denoting the object.

For speed estimation it is necessary to have a distance information to use time data obtained from the video and, if the tracking has been done correctly, we will be able to obtain an estimated speed that is close to the real one. Other methods could be done using geographic data or an estimated Field of View calculation, but they would be either too complicated or would end up being an estimate anyway.

After a brief explanation of the tasks that are managed in the system itself, although it will be explained later on how these actions have been applied within the system and going a little deeper into them, we will go on to explain concepts that have been used for the algorithm used in the project.

The binarisation or thresholding of an image consists of establishing a threshold whereby if a pixel is greater in terms of colour than that limit point, the maximum value established will be set and if it is below the threshold, it will be 0. There are also other methods such as Otsu, which does not need to establish a threshold but rather determine the optimum threshold for the image. It does this by assuming that there are only two types of pixels: those at the front of the image and those at the back of the image. With this it calculates the grey scale histogram of the intensity of the pixels in the image, separating into two peaks and by this calculates the optimal threshold. [38]

The KNN algorithm [39] is an algorithm that takes its name from the English K-Nearest neighbours, which belongs to the type of algorithms called supervised learning algorithms. What does this mean? Well, they are algorithms that receive a set of data that

are labelled with the output values on which they can be trained and through this define a prediction model.

Once this is known, the algorithm works as follows:

- Establish the number of K neighbours.
- Calculate the possible distance between them (Euclidean, Manhattan...).
- Take the K nearest neighbours depending on the calculated distance.
- Among the K neighbours, count the resulting number of each label or category of neighbours.
- Attribute a new point to the category most present among the nearest neighbours.

A background subtractor [40] is one of the most popular approaches to object detection in images and works by comparing parts of the video or frame that are moving relative to the background or in front of it.

This method is used to find objects in the front of the video so that it compares them with frames where no objects are present, in this way it will find the differences between them and create a distance matrix.

It will compare the difference of these two situations and the threshold that is set, if it is higher, it will be determined as a moving object.

A.3 SOCIOECONOMIC AND LEGAL ENVIRONMENT

This section will cover both the legal framework of the project, explaining its laws and how they can influence the project, and the socio-economic environment and how it can influence society.

A.3.1 SOCIOECONOMIC ENVIROMENT

The aim of the project, if it were to be implemented in a real environment and not only in a simulated environment, is to monitor people's driving and control it so that there are fewer violations and accidents. Therefore, taking this into account, it is expected that after applying the use of drones on roads, this type of incidents could be considerably reduced. Furthermore, this method could cover a much greater amount of ground than using a helicopter or a basic police car.

According to a DGT report issued on 7 July 2021, it is said that the drones present in Spanish territory for traffic tasks have been in flight for some 500 hours since 2018 and have detected more than 600 infractions, having become one of the most effective methods for infractions such as the use of mobile phones at the wheel, incorrect use of seat belts and child restraint systems, as well as overtaking without respecting the distance

to cyclists. Therefore, it can be assumed that they have been a great addition to the Dirección General de Tráfico. [52]

On the economic side, it should be noted that according to an article by the Lisa Institute [53], using drones in emergency situations is 12 times more cost-effective than using conventional helicopters and manned aircraft. This is an advantage when compared to other aerial vehicles that carry out road monitoring work.

And, with the environment in mind, it should be noted that as these are non-petrol, battery-powered vehicles, the energy needed to use them could come from less polluting non-petroleum sources such as renewable energies like solar panels or wind power, which is another possible advantage over helicopters.

Although it is true that batteries are normally made of lithium, which is not a particularly healthy material for the environment, it is a better solution than the use of compounds derived from petroleum, and if these renewable energies were improved to obtain a greater quantity or if they were more widely used, it would be even more optimal if possible.

According to an article in El Confidencial [54], a Volvo study has concluded that the manufacture of an electric car is much more harmful to the environment because of the CO₂ emissions cost in the manufacture of lithium-ion batteries, which pollutes more than the manufacture of normal petrol engines, but it also details that, over time, this emissions deficiency is reduced and ends up reducing its emissions with respect to its petrol counterpart. Although it would have to be seen how the energy is sourced and whether the methods use a high percentage of non-renewable energy, otherwise the difference would not be a detail to highlight.

Although the article above relates the difference in cars, the cost of manufacturing drone batteries would be the same in percentage terms.

A.3.2 LEGAL ENVIROMENT

Currently in Spain, national regulations are no longer in force as of 1 January 2022, but with exceptions for model aircraft clubs and associations that will be in place until 1 January 2023 when European regulations will be fully adopted.

However, it is worth noting one type of activity that does not follow the European rules, namely non-EASA activities [47]. This group is not required to register the activities they carry out in the AESEA Operators' Register, although they must have the relevant qualifications to fly a drone and follow all safety guidelines.

As European legislation is followed in Spain, it is necessary to explain it in order to take into account the existing restrictions on the use of this type of vehicle.

The open category includes all flights with a low level of risk, with several clear restrictions:

- The overflight of groups of persons is prohibited.
- The transport or dropping of dangerous materials or goods is not permitted.
- Autonomous operations are not allowed.

There are also the following requirements to be able to fly in this category:

- You must be at least 16 years old.
- The drone operator must be registered in the AESEA Operator Register.
- Pass theoretical training and a test to certify that they have the necessary knowledge to fly a drone.
- Keep the drone in line of sight at all times.
- The maximum permitted flight altitude is 120 metres.
- The maximum take-off mass of the drone is less than 25 kilos.

Finally, there are three subcategories within this category depending on the weight A1 (less than 250 grams), A2 (less than 4 kilograms) and A3 (less than 25 kilograms).

The specific category includes flights that cannot be included in the open category because of their risk. The characteristics for a flight to be considered in this category are as follows:

- Flights beyond line of sight.
- Operations above 120 metres altitude.
- Drones weighing more than 25 kilograms.
- Urban flights with drones over 4 kilograms.
- Dropping of materials.
- Flying over large groups of people.

In addition, age requirements must be met, as well as being registered in the Operators' Register and a study on risky operations.

And finally, the certified category covers the following:

- Drones certified under EU Delegated Regulation 2019/945.
- When flying over groups of people with a drone with a wingspan of more than 3 metres.
- When flying over crowds of people; transporting dangerous goods with high risk in case of an accident; or when it involves the transport of people.

If the safety study mentioned above indicates the need for certification of the UAS and the operator and obtaining a pilot's licence.

Another problem that would arise when carrying it out in a real environment would be the possible violation of the Organic Law on Data Protection [49]. In order to carry out this project, it would be necessary to record roads, cars with their number plates, as well

as the possibility of faces of both adults and minors appearing, which could violate this law by possessing these videos without the authorisation of the people who appear in them and without being an authority of the Spanish State.

And, to conclude this chapter, we should ask ourselves whether Spanish law allows for the decision making by an Artificial Intelligence algorithm to end in a fine.

Although drones can detect infractions such as drivers not wearing seat belts or using a mobile phone at the wheel, this detection is not always reliable, as it can fail for various reasons, for example if the driver is wearing a seat belt-coloured garment, which could circumvent the system.

In any case, no matter how advanced the algorithm used is, all possible positives in terms of infringement must currently be manually checked for these possible errors that may arise. In other words, Artificial Intelligence cannot fine directly, but must first be reviewed by the competent authorities. [50]

Although the law does include the use of Artificial Intelligence, it is usually in cases to speed up the collection of relevant information to process the complaint to the citizen. Algorithms are even allowed for the drafting of the proposed resolution, but only in cases where there is no opposition from the citizen.

A.4 SOLUTION DESIGN

In this section, an explanation of all the parts present in the system will be made using flow diagrams and images, although not of the final system, for this purpose.

As the system is divided into several sections, such as the patrolling movement of the vehicles or the computer vision algorithm itself, the explanation of the design will be divided into these parts.

A.4.1 DRONE MISSION DESIGN

In order to carry out the drone's mission, the point of origin of coordinates must be taken into account, which is found in GPS coordinates, and from this the NEDYaw coordinates have been used, which refers to how much it moves North, East and Down and how much it rotates in terms of its direction. By means of this, it can be made to rise to avoid possible obstacles and move anywhere in the simulated environment, although in order for it to move correctly and not join one movement with another, it is necessary to wait a few seconds for the drone to be able to do so.

Once the drone has moved to the desired area and the camera has been set to the required angle, it will start recording a section of the road using the drone's recording function and will stop recording when it is considered necessary.

A.4.2 DESIGN OF CAR TASKS

In order to implement the system, it is necessary that at least the cars move in one direction and thus be able to estimate the speed of the vehicle.

To achieve this functionality, we have followed the basic patrol model of videogames where a path is established before execution by creating an Unreal Engine actor, which in our case has been called VehiclePath, and adding a 'spline' component to it to function as points of a path to follow, as can be seen in Figure 51.

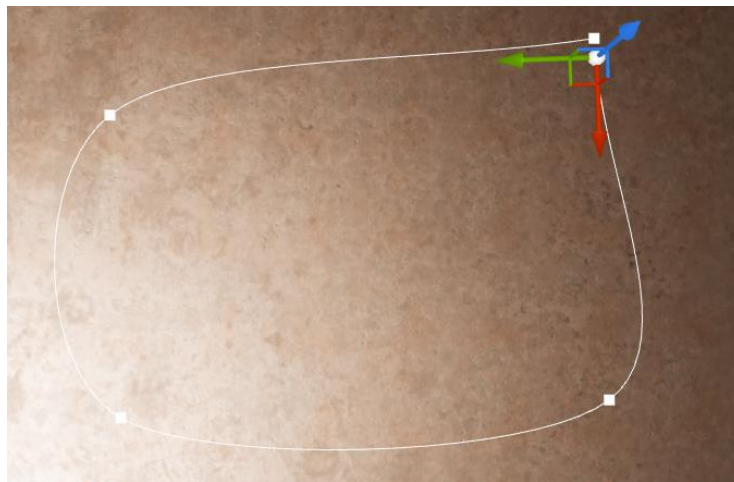


Figura 51. Path to follow example

With these pre-established points, it is possible to generate any type of path and have the car drive along them. It travels them using the function that has been created called GetPath, which is a function that takes the array of points of the path and gives the direction of these points to the vehicle so that it can go to their location, all of this within the 'Tick' event that is executed in each frame of the game.

The GetPath function performs a series of mathematical calculations to know the position of each point with respect to where the car itself is. These calculations are as follows: First it takes the array of the path and returns the first element of it making a copy of this, when it passes through this point the path will not be the first in the list, and through this element, which is a location within the world, the closest tangent vector to this position is obtained to normalise this vector and scale it later with a tolerance of 0.0001, i.e. the result of the scaling of the vector can vary that value. With the vector already scaled, it is added with the vector obtained from the position of the vehicle, which is obtained by means of the function GetActorLocation and as the actor in this case is the vehicle itself, it returns its position in the form of a vector. This vector added is used to obtain the point of the curve closest to this location, which will be taken as the point to advance, and taking the position of the vehicle and this target position you get the rotation to be performed towards that point, if you join it with the GetActorRotation function, which takes the current rotation of the vehicle, by normalising it with these two vectors and the

Unreal Engine delta function of the rotation you get the direction to where it has to move (Yaw) and this rotation angle is scaled with respect to 90 and -90 taking from 1.0 and -1.0, that is to say in an initial point of rotation 0.5 when passing it through this function the new value would be 45, and it is passed finally as output parameter so that it takes that direction in concrete.

To conclude with the functionalities of the movement of the car, a functionality has been developed to maintain a speed that is proposed over time, in order to have another method to check the speed apart from the CSV that is written, and also to check that this speed of the log file is correct. The velocity is calculated by using the GetVelocity function in Unreal Engine which returns cm/s and then using some calculations you can convert it to km/h in the XY vector to know the correct velocity.

This velocity in km/h can be passed from a general vector to the XY vector in two ways, both of which have been implemented to evaluate its operation. The first is to check the length of the vector, the longer the velocity vector the higher the velocity, and multiply it by 0.036 to convert it to km/h and the other method is to use the following equation: $\sqrt{x^2 + y^2}$, with x and y being the two velocity vectors in those planes, to get the value of the vector in XY and again it is passed to km/h.

Then once you have these two things you compare the speed of the car itself with one that is set as a limit and when it is below it, the brake is turned off, setting it to 0 and setting the acceleration to 1.0 (the maximum possible) and if it exceeds it, vice versa.

Another functionality presented by the vehicles is the writing of vehicle information at runtime into a CSV file. This information is as follows: Position of the car in X, Y and Z with respect to the position of the drone because to calculate the coordinates in the real world you should have these coordinates, the timestamp of when the information was taken, the vehicle tag, the position of the drone and the speed of the car in all planes (X, Y, Z, XY).

A custom blueprint is used, as there is no way to write to a CSV file within the Unreal Engine itself and it had to be implemented in C++. To call this function, a cyclic counter has been used, which is executed every second (or as long as it is considered necessary) and is written in each line of the file.

To finish with the functionalities of the cars, the car is able to wait for a certain amount of time before it is ready to move forward or collect information so that the drone has time to reach the specific place before the car moves forward.

A.4.3 DESIGN OF THE SPEED ESTIMATION ALGORITHM

This design has been divided into three sections: vehicle detection, vehicle tracking and speed estimation.

In terms of detection, the video resulting from the recording will be taken and a computer vision algorithm will be applied to detect the vehicles, their tracking and an estimation of the speed at which they are moving. Before proceeding, it is worth noting that whenever functions are mentioned, they are functions of OpenCV itself. And it is also good to note that a resolution change of the video is performed in order to be able to check the three resulting windows at the same time (mask, region of interest and video).

For the detection we have used an environment subtractor that uses the KNN algorithm explained in the section on the state of the art, and by means of the detector resulting from this an environment mask is computed only on the region of interest, which in our case is the road, since trees and others are omitted. From this mask, the threshold function is applied, which binarises the image, and by means of the findContours function, the contours of the objects in this binarised image are found. By going through all the contours of the frame, the area of the contour is obtained using the function contourArea and depending on whether the contour exceeds a specified area, it will be considered an object and its bounding box will be saved using the function boundingRect, which returns the upper left (x,y) point of the square, its width and height, to later save this bounding box in a list.

For tracking, taking all the bounding boxes obtained by detection, the centroid of the rectangle is calculated. That is, by the following calculation: $\frac{(x+x+w)}{2}$ for the X-plane and $\frac{(y+y+h)}{2}$ for the Y-plane, the two fractions being ground functions.

If that object has not been detected, which is handled with a boolean, it is set as a Python object and is added to a list where the bounding box and the id of each object will be found, as well as adding one to the ids of the objects to have a record of those that have been present. Once the object has been detected, it is checked if the Euclidean distance, which is calculated with the following formula: $\sqrt{(x - x_{Previous})^2 + (y - y_{Previous})^2}$ between the current centroid and the previous one exceeds a specific threshold, if it does not exceed it will be considered as the same object and will be added to the list commented above, with this the following information will be written in the CSV each time it does not exceed the threshold: the id of the object, its centroid in the frame, a timestamp and if it has exceeded a point in the image, the use of which will be explained later.

When this process is finished, the list of bounding boxes and ids will be scrolled through, and the image will show the bounding box in the required position and a text with the following format: Obj: ID of the object.

And finally, the speed estimation, this has been calculated with the physical formula *velocidad = espacio/tiempo* where the space is an estimated distance through elements of the image such as a Cesium bench, which, although it is squashed, can be used to know where it is located using Google Maps, or the traffic signs which, if they are in 3D. From this, a more or less useful estimated distance can be obtained for use in the calculation. The time used in the formula is the difference between the first time the object is recognised in the video processing and the first time a 1 is found in the CSV

associated with the ID of that object, as this will mean that the centroid has passed through the line used in the video that emulates the previously estimated distance. This is used to calculate the speed at which the vehicle is travelling.

A.5 PLANNING AND BUDGET

This section will discuss the planning of the entire project, explaining each section and showing the budget that would be used if this project were to be carried out.

A.5.1 PLANNING

Although they will later be indicated in the Gantt chart and detailed in terms of the number of days needed for each one, it is necessary to explain the phases of the project so that they are clear when it comes to showing them in the following points.

- **Contact and formulation of the proposal:** This phase includes all the time required to contact the tutor for the general topic found on the university's Final Degree Project board and the discussion carried out to determine the proposed objectives.
- **Initial study of the technologies used:** This refers to the initial phase of the project, from the installation of everything necessary for its implementation and verification of its correct operation to small tests of MavSDK functions such as flight tests or photography in order to become familiar with the relevant technology.
- **Development:** This is the phase in which the project is developed to realise all the proposed functionalities.
- **Research:** This refers to the whole process of researching and searching for concepts in order to be able to explain everything correctly in the memory, as well as algorithms, functions, how the technology to be used is used, etc. Thus, as the research also entails the search for information to be explained in this document, this phase will be intermingled with both the development phase and the memory phase.
- **Memory:** This is the time that has been dedicated to the realisation of this document and all the graphics and explanations necessary to correctly describe the project that has been realised.
- **Revision:** This refers to the revision phase of the report where the tutor is given a copy of it and points out possible errors or improvements that can be carried out and subsequently fixed.

The planning of the project has been as shown in the following Gantt chart:

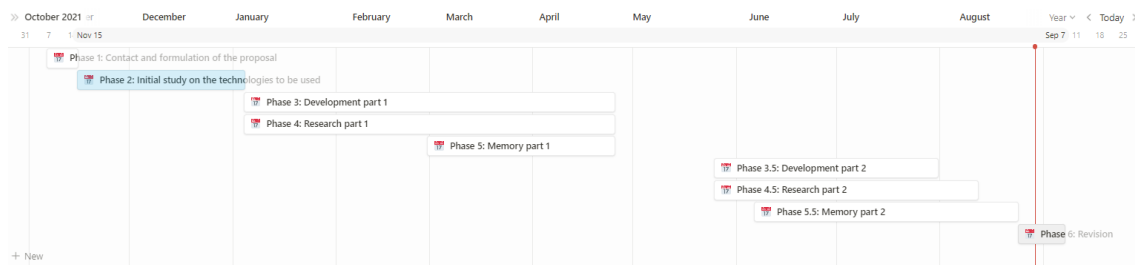


Tabla 17. Gantt chart of actual planning

The table from which the diagram has been made will be shown for clarity:

Milestone	Dates	Σ Days
Phase 1: Contact and formulation of the proposal	November 6, 2021 → November 15, 2021	9
Phase 2: Initial study on the technologies to be used	November 15, 2021 → January 4, 2022	50
Phase 3: Development part 1	January 4, 2022 → April 25, 2022	111
Phase 3.5: Development part 2	May 25, 2022 → July 31, 2022	67
Phase 4: Research part 1	January 4, 2022 → April 25, 2022	111
Phase 4.5: Research part 2	May 25, 2022 → August 12, 2022	79
Phase 5: Memory part 1	February 28, 2022 → April 25, 2022	56
Phase 5.5: Memory part 2	June 6, 2022 → August 24, 2022	79
Phase 6: Revision	August 24, 2022 → September 7, 2022	14

Tabla 18. Actual project planning

In other words, the total number of days spent on the project was a total of **275 days**, taking into account the days from the start of the project on 6 November 2021 to the end of the first half on 25 April, including the days of the second half from 25 May to 7 September, giving that total number of days.

A.5.2 BUDGET

Starting with the material costs, as the licensing of all software used during the entire project is free of charge, only the hardware costs will be included, which are presented in the following table:

Component	Model	Cost (euros)
Processor	AMD Ryzen 5 2600	165,50
RAM Memory	16 GB DDR4	148,11

Power supply	EVGA 600 W1	54,89
GPU	GeForce GTX 1050 TI	174,90
Hard disk 1	SSD 120 GBs	29,04
Hard disk 2	HDD 1 TB	45,04
Monitor 1	BenQ GW2470H	125,60
Monitor 2	AOC G2590FX	199,00
Motherboard	ASUS Prime B350-PLUS	88,88
Keyboard	Teclado Mecánico AKKO	100,00
Mouse	Logitech G203	21,99
Total		1152,95

Tabla 19. Material costs.

Also, to be considered are the costs in terms of salaries that could be incurred in case of the realisation of the project, as the project has only been conducted by one programmer, these costs will be estimated as those of a single employee. Taking as a reference an engineer's salary according to the Resolution of 10 March of the Directorate General of Traffic with respect to the salary tables, which presents a salary of 1,832.07 € [45] and taking this as a full-time salary is 11.45 € per hour and as has been reported in the final planning of the project, a total of 275 days have been worked, at an average of 2.5 hours per day, as it has been reported in the final planning of the project. 5 hours per day, as there have been days with more than ten hours of work, but others with more relaxed working days and therefore this is considered a good average of hours worked per day, and so the cost is shown in the following table:

Cost per hour (euros)	11,45
Hours spend	687,5
Total cost (euros)	7.871,88

Tabla 20. Personal costs

Finally, a table will be shown that explains the overall costs of the entire project, combining the costs mentioned in the previous sections with other types of costs such as indirect costs, profit margin and value added tax, which in the case of software and services is 21% [46]. All of this is set out in the following table:

Concept	Cost (euros)
Direct costs	9024,83
Overhead costs (10%)	902,48
Profit margin (25%)	2256,21
IVA (21%)	1895,22
Total costs	14078,74

Tabla 21. Total costs

Total: FOURTEEN THOUSAND AND SEVENTY-EIGHT EUROS AND SEVENTY-FOUR CENTS.

A.6 RESULTS

It is necessary to indicate the initial state of the system and where the drone starts from. For the tests, the Cesium map of the Colmenarejo campus of the Carlos III University of Madrid has been used, with the following initial position: (Latitude: 40.544289, Longitude: -4.012101), which is shown in Figure 37 and is where the drone starts to move to the road on the left to begin recording.

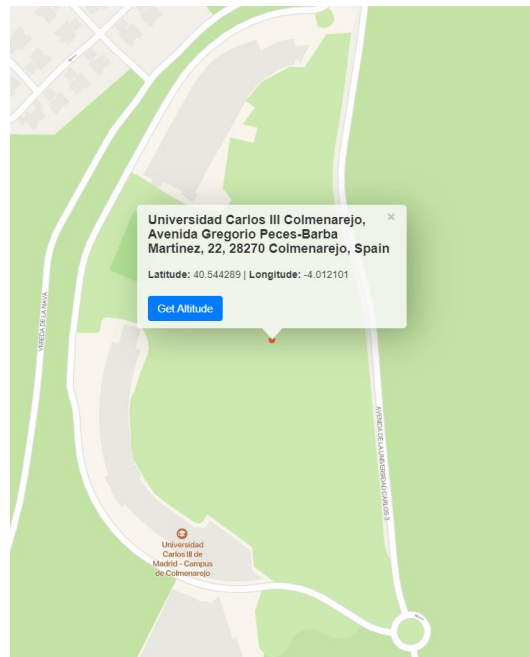


Figura 52. Initial drone location

To finish with the evaluation section of the algorithm, a comparison will be made between the real average speeds and the estimated average speeds in each case, highlighting the margin of error in the algorithm and what has occurred with the noise in the videos.

Therefore, the following graph shows a comparison between the two speeds discussed:

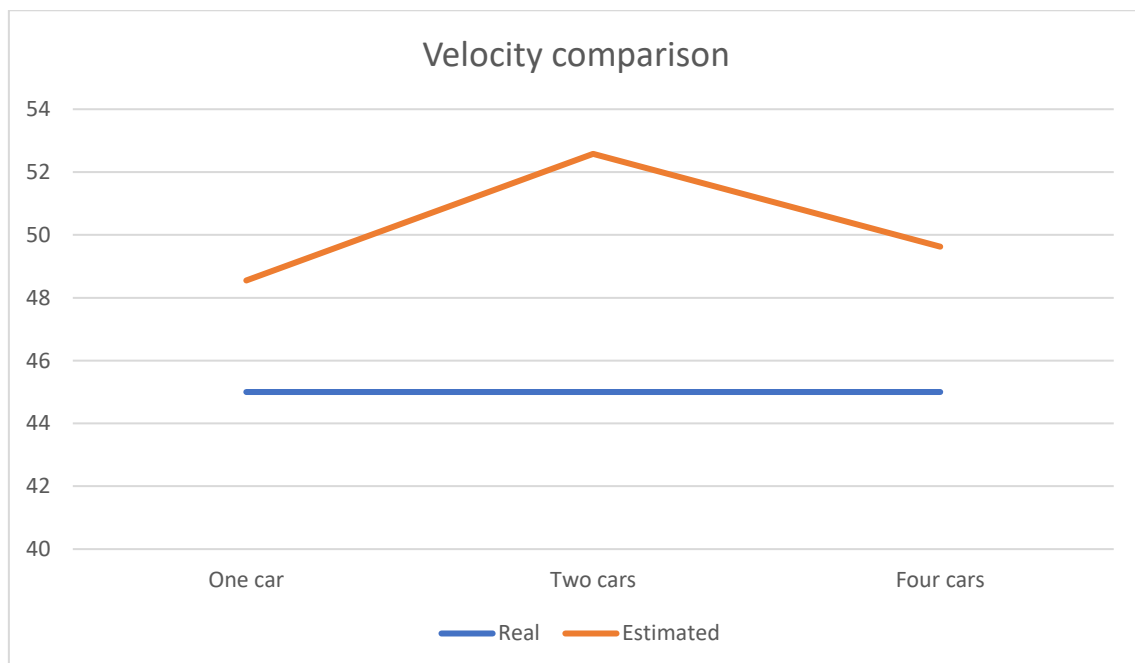


Figura 53. Speed comparison chart.

From this graph it can be seen that each of the test cases have the following margins of error compared to the actual speed that has been proposed as the limit, which in all cases

has been 45 km/h, which is known because it has been checked in the data shown in the CSV and because the system is designed for vehicles to maintain a fixed speed:

One car: 7.89% One car: 7.89% Two cars: 16.85% Two cars: 16.85

Two cars: 16.85

Four cars: 10.3%.

What you get from this is that it is not an exact algorithm, but you get results that are sufficiently close to the original data to be considered valid. However, it should be clarified that, although it may seem that increasing the number of cars does not worsen the results because it has a better margin of error than two cars, this is not true, since it has been observed that the more cars there are, the more noise there is in each frame, and this can result in errors in the tracking.

Another aspect that has been observed is that if the vehicles are very close together it can be considered as a single detected object and therefore it is better to leave some separation between them in the tests. In addition, it has been recognised that the cars in the lane closest to the camera, i.e., the lane that goes from left to right from the horizontal camera, have worse results, in tests with the drone above the road the same thing happens, and this is due to the noise that the simulator terrain must generate close to that lane.

A.7 CONCLUSIONS

Finally, we will draw some conclusions about the project in general and some personal conclusions about the whole experience while doing it.

A.7.1 PROJECT CONCLUSIONS

As for the conclusions we can draw from everything developed in the project, we can deduce that previous knowledge has been obtained both of computer vision algorithms and of the general functioning of the Unreal Engine and of drones as unmanned vehicles, including all their aspects.

Regarding the results obtained in the tests, as shown above in the evaluation section, they are satisfactory enough to be considered valid, but the noise generated by the images making the IDs not only correspond to the cars suggests that the system is not scalable to a busy road.

Therefore, to conclude this section, the resulting system is valid and functional, and from it, it has been possible to explain and show some concepts that have served both the author of this, to better understand the development and to be able to do it as well as possible, as well as the possible reader so that its reading is sufficiently understandable even without knowing anything about any field related to those present in the project.

A.7.2 PERSONAL CONCLUSIONS

This section is going to be written from a first-person point of view, as it is, after all, the author's opinion.

I would like to start by putting the feelings I have had during the development of the work. To begin with, I think that with its ups and downs the project has been enjoyable, there have been parts that have been more pleasant to do and that I have found more satisfying and others less so, but looking at it as a whole I think I have ended up enjoying it.

Also, it is necessary to mention that I had never touched anything related to Artificial Intelligence at this point, I only had knowledge of algorithms like the A star and others but very little in general and I think that if I had had some more base I could have been less frustrated during the realization of this, because yes, in the end there was a point in which I got frustrated in its development, after many attempts trying algorithms and others to have an optimal detection, but once passed this point everything was something more enjoyable.

Although the Artificial Intelligence part was a bit more difficult, the part of using the Unreal Engine programming with the 'blueprints' and preparing the simulated environment was, without a doubt, the part I enjoyed the most because I have always liked videogames and having a little look at the nodes that Unreal uses for programming them was quite enjoyable.

Another point to take into account is that according to the way the basic programming of the university is done with regard to the internships and the subjects of the last term, it is very difficult to get enough time to finish it for the June exam, unless you get an internship in the previous summer, it was very difficult for me to combine everything and, therefore, it had to be delayed to September. That's why I think this is a point that could be improved.

Finally, I would like to say that, although there have been complications in the development, I have enjoyed the way to complete the work and that I have learned a lot about areas and concepts that are very interesting.

ANEXO B. REQUISITOS

B.1 REQUISITOS FUNCIONALES

A continuación, se van a mostrar los requisitos de sistema funcionales:

ID	RF-01				
Título	Conexión del dron con el computador.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El dron debe poder conectarse con el computador mediante MAVLink y PX4.				

Tabla 22. Requisito funcional 01

ID	RF-02				
Título	Comunicación del dron con Unreal Engine.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El dron debe poder recibir las acciones y tareas del código correspondiente y realizarlas en el mundo del Unreal Engine.				

Tabla 23. Requisito funcional 02

ID	RF-03				
Título	Movimiento del dron a una zona específica.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El dron debe ser capaz de moverse a una zona descrita anteriormente sin ningún problema ni choque.				

Tabla 24. Requisito funcional 03

ID	RF-04				
Título	Grabación de imágenes a través del dron.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El dron grabará imágenes que están dentro del campo de visión de su cámara al pulsar el botón 'R' mientras se encuentra en ejecución.				

Tabla 25. Requisito funcional 04

ID	RF-05				
Título	Unión de frames para conseguir un video.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El sistema debe poder unir los frames obtenidos de la grabación y realizar un video ordenado resultante de esos frames.				

Tabla 26. Requisito funcional 05

ID	RF-06				
Título	Vehículos en la simulación.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	Dentro de la simulación se debe poder introducir vehículos en distintas posiciones del mapa.				

Tabla 27. Requisito funcional 06

ID	RF-07				
Título	Aceleración del vehículo.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	Los vehículos deben poder acelerar en un rango de 0.0 a 1.0 dentro de Unreal Engine.				

Tabla 28. Requisito funcional 07

ID	RF-08				
Título	Freno del vehículo.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	Los vehículos tienen que frenar en situaciones específicas.				

Tabla 29. Requisito funcional 08

ID	RF-09				
Título	Control de la velocidad del vehículo.				
Tipo	Funcional	Prioridad	Media	Verificabilidad	Sí
Descripción	Cada vehículo debe ser capaz de controlar su velocidad frenando y acelerando, dependiendo de si ha superado o no la velocidad límite propuesta.				

Tabla 30. Requisito funcional 09

ID	RF-10				
Título	Seguimiento de un camino por parte del vehículo.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	Cada vehículo debe poder seguir de manera autónoma un camino insertado en Unreal Engine mediante la componente 'Spline' del mismo.				

Tabla 31. Requisito funcional 10

ID	RF-11				
Título	Espera en el inicio del movimiento del vehículo.				
Tipo	Funcional	Prioridad	Baja	Verificabilidad	Sí
Descripción	Un vehículo debe poder esperar un periodo de tiempo descrito antes de la ejecución y esperar ese periodo antes de realizar ninguna acción.				

Tabla 32. Requisito funcional 11

ID	RF-12				
Título	Escritura de información en un archivo CSV.				
Tipo	Funcional	Prioridad	Media	Verificabilidad	Sí
Descripción	El sistema debe poder escribir información de los vehículos en un archivo formato CSV para poder analizar esta posteriormente.				

Tabla 33. Requisito funcional 12

ID	RF-13				
Título	Localización del vehículo.				
Tipo	Funcional	Prioridad	Media	Verificabilidad	Sí
Descripción	Se debe poder obtener la información relativa a la ubicación del vehículo en los ejes X, Y y Z en el mundo de manera periódica cada segundo y que esta sea escrita en el CSV.				

Tabla 34. Requisito funcional 13

ID	RF-14				
Título	Marca de tiempo del vehículo.				
Tipo	Funcional	Prioridad	Media	Verificabilidad	Sí
Descripción	Se debe poder obtener la información relativa a la marca del tiempo del vehículo en el momento de ejecución de manera periódica cada segundo y que esta sea escrita en el CSV.				

Tabla 35. Requisito funcional 14

ID	RF-15				
Título	Etiqueta del vehículo.				
Tipo	Funcional	Prioridad	Baja	Verificabilidad	Sí
Descripción	Para poder diferenciar los distintos vehículos, se debe poder obtener la información relativa a la etiqueta señalizada al vehículo en cuestión de manera periódica cada segundo y que esta sea escrita en el CSV.				

Tabla 36. Requisito funcional 15

ID	RF-16				
Título	Posición del dron.				
Tipo	Funcional	Prioridad	Media	Verificabilidad	Sí
Descripción	Se debe poder obtener la información relativa a la ubicación del dron en el mundo de manera periódica cada segundo y que esta sea escrita en el CSV.				

Tabla 37. Requisito funcional 16

ID	RF-17				
Título	Velocidad del vehículo en cada eje.				
Tipo	Funcional	Prioridad	Baja	Verificabilidad	Sí
Descripción	Se debe poder obtener la información relativa a la velocidad del vehículo en los ejes X, Y y Z de manera periódica cada segundo y que esta sea escrita en el CSV.				

Tabla 38. Requisito funcional 17

ID	RF-18				
Título	Velocidad del vehículo en el eje XY.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	Se debe poder obtener la información relativa a la velocidad del vehículo en el eje XY, la velocidad “real” en kilómetros del mismo, de manera periódica cada segundo y que esta sea escrita en el CSV.				

Tabla 39. Requisito funcional 18

ID	RF-19				
Título	Detección de vehículos.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El sistema debe poder detectar vehículos en una grabación y reconocerlos como objetos asignándoles un Id.				

Tabla 40. Requisito funcional 19

ID	RF-20				
Título	Tracking de los vehículos.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El sistema debe poder realizar un tracking de los vehículos en una grabación y hacer un seguimiento de los mismos a través del vídeo manteniendo la misma ID a lo largo de todo el video.				

Tabla 41. Requisito funcional 20

ID	RF-21				
Título	Monitorización del movimiento del vehículo.				
Tipo	Funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El sistema debe poder notar cuando un vehículo de la grabación cruza una zona del vídeo con una marca de tiempo.				

Tabla 42. Requisito funcional 21

B.2 REQUISITOS NO FUNCIONALES

Y para terminar con la enumeración de requisitos, ahora se describirán los requisitos de sistema no funcionales:

ID	RN-01				
Título	Funcionamiento en WSL.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El sistema debe ser ejecutado en WSL (Windows Subsystem for Linux).				

Tabla 43. Requisito no funcional 01

ID	RN-02				
Título	Tiempo para correr el sistema de simulación.				
Tipo	No funcional	Prioridad	Baja	Verificabilidad	Sí
Descripción	El tiempo necesario para correr todo el sistema, tanto el Visual Studio Code con todo el código del dron en WSL como Unreal Engine y todo lo necesario para el sistema no debe ser superior a 1 minuto en un computador de gama media-alta.				

Tabla 44. Requisito no funcional 02

ID	RN-03				
Título	Compatibilidad con PX4.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El sistema debe ser compatible con el software PX4 funcionando como controlador de vuelo.				

Tabla 45. Requisito no funcional 03

ID	RN-04				
Título	Comunicación con el computador con MAVLink.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El dron debe poder conectarse con el computador mediante MAVLink para mantener una comunicación constante.				

Tabla 46. Requisito no funcional 04

ID	RN-05				
Título	Calidad de la cámara.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	La cámara del dron debe tener mínimo una calidad de 480p para poder realizar el procesado de imágenes de manera correcta.				

Tabla 47. Requisito no funcional 05

ID	RN-06				
Título	Tiempo de unión de frames.				
Tipo	No funcional	Prioridad	Baja	Verificabilidad	Sí
Descripción	El tiempo necesario para unir los frames obtenidos mediante la cámara del dron no debe ser superior a 1 segundo si el video resultante es de 6 segundos.				

Tabla 48. Requisito no funcional 06

ID	RN-07				
Título	Tiempo de procesado de vídeo.				
Tipo	No funcional	Prioridad	Baja	Verificabilidad	Sí
Descripción	El tiempo necesario para procesar un vídeo de 5 segundos con toda la información necesaria del mismo (objetos, tracking de estos, timestamp) y mostrarlo no debe ser superior a 1 segundo en un vídeo de 6 segundos.				

Tabla 49. Requisito no funcional 07

ID	RN-08				
Título	Lenguaje de programación.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El lenguaje de programación utilizado en casi la totalidad del sistema será Python.				

Tabla 50. Requisito no funcional 08

ID	RN-09				
Título	Compatibilidad de coordenadas.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El sistema debe ser compatible con coordenadas GPS (latitud y longitud) y con coordenadas NED en el dron (Norte, Este, Abajo).				

Tabla 51. Requisito no funcional 09

ID	RN-10				
Título	Sistema de backups.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El sistema debe poseer un sistema de backups usando Git y Drive para ello.				

Tabla 52. Requisito no funcional 10

ID	RN-11				
Título	Código abierto.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	Todos los componentes del sistema deben haber sido realizados con herramientas de código abierto y por ende el sistema final debe ser de código abierto.				

Tabla 53. Requisito no funcional 11

ID	RN-12				
Título	Entorno en Unreal Engine.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El entorno de simulación debe ser realizado en el motor de juego Unreal Engine.				

Tabla 54. Requisito no funcional 12

ID	RN-13				
Título	Simulación con AirSim.				
Tipo	No funcional	Prioridad	Alta	Verificabilidad	Sí
Descripción	El simulador de drones utilizado será AirSim.				

Tabla 55. Requisito no funcional 13

ID	RN-14				
Título	Plugin de Cesium.				
Tipo	No funcional	Prioridad	Media	Verificabilidad	Sí
Descripción	Para aportar mayor realismo, el sistema utilizará Cesium con Unreal Engine en su mapa simulado.				

Tabla 56. Requisito no funcional 14