UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Algoritmos de detección en sistemas MIMO masivo para comunicaciones digitales con drones.

Miguel Fernández Dasí

Tutor: Alberto González Salvador

Cotutor: Mª Ángeles Simarro Haro

Trabajo Fin de Máster presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Máster en Ingeniería Telecomunicación

Curso 2020-21

Valencia, 12 de septiembre de 2021

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO. ES-CV- 000030

Resumen

Hoy en día el uso de vehículos aéreos no tripulados, generalmente referidos como drones, para la realización de tareas que entrañan un riesgo para un humano es algo que se encuentra muy cerca de lo cotidiano. Con la explosión de su uso con fines lúdicos, la oferta de este tipo de vehículos se ha disparado, tanto es así que hoy en día existen regulaciones que prohíben volar drones en zonas urbanas.

Sin embargo, desde principios de la década de 2010, se han utilizado para realizar todo tipo de carreras alrededor del mundo. Con el paso de los años se han creado ligas y federaciones que han ayudado a popularizar este deporte que siempre ha estado en crecimiento desde sus inicios. En la actualidad, las carreras sufren el problema de seguir dependiendo de tecnología analógica para transmitir la imagen capturada por los drones, siendo el rango y cantidad de participantes los puntos en los que más sufren.

Es por esto que, es necesaria la inclusión de tecnologías digitales en las carreras de drones para mejorar la calidad de este deporte y, así, aumentar su difusión.

Resum

Hui dia l'ús de vehicles aeris no tripulats, generalment referits com a drons, per a la realització de tasques que comporten un risc per a un humà és una cosa que es troba molt prop del quotidià. Amb l'explosió del seu ús amb finalitats lúdics, l'oferta d'aquest tipus de vehicles s'ha disparat, tant és així que hui dia existeixen regulacions que prohibeixen volar drons en zones urbanes.

No obstant això, des de principis de la dècada de 2010, s'han utilitzat per a realitzar tot tipus de carreres al voltant del món. Amb el pas dels anys s'han creat lligues i federacions que han ajudat a popularitzar aquest esport que sempre ha estat en creixement des dels seus inicis. En l'actualitat, les carreres pateixen el problema de continuar depenent de tecnologia analògica per a transmetre la imatge capturada pels drons, sent el rang i quantitat de participants els punts en els quals més pateixen.

És per això que, és necessària la inclusió de tecnologies digitals en les carreres de drons per a millorar la qualitat d'aquest esport i, així, augmentar la seua difusió.

Abstract

Nowadays, the use of unknown aerial vehicles, normally referred as drones, to do tasks that will be risky for humans is becoming less exceptional. With the increase of its use for recreational purposes, the supply of this type of vehicle has skyrocketed, so much so that today there are regulations that prohibit flying drones in urban areas.

However, since the early 2010s, drones have been used to run all kinds of races around the world. Over the years, leagues and federations have been created that have helped popularize this sport that has always been growing since its origins. Nowadays, races suffer from the problem of relying on analogue technology to transmit the video signal captured by drones, with the range and number of participants being the points where they suffer the most.

Therefore, the inclusion of digital technologies in drone racing is necessary to improve the quality of this sport and thus increase its diffusion.

Índice

Capítulo 1. Introducción, Motivación y Objetivos 3

Capítulo 2. Estado del arte 9

2.1 VTX Analógico y Digital 9

2.2 Gafas FPV 11

2.3 Antenas utilizadas en transmisión y recepción 12

2.4 Ventajas y desventajas introducidas por mMIMO 14

Capítulo 3. Evaluación de los Costes del Proyecto 16

3.1 Costes económicos del proyecto 16

3.2 Costes temporales del proyecto 17

Capítulo 4. Algoritmos de Detección Empleados 20

4.1 Modelo del Sistema 20

4.2 Algoritmos de detección seleccionados 21

4.2.1 Optimized Coordinate Descent-based BOX Equalization 21

4.2.2 ADMM-based Infinity Norm 24

4.2.3 Successive Over Relaxation Method 25

4.2.4 Sequential Likelihood Ascent Search 26

Capítulo 5. Pruebas realizadas 30

5.1 Condiciones de las pruebas 30

5.2 Estudio del BER de los algoritmos 34

5.2.1 Simulaciones OCD-BOX 34

5.2.2 Simulaciones ADMIN 37

5.2.3 Simulaciones SOR 40

5.2.4 Simulaciones SLAS 44

5.3 Prueba de transmisión de video 46

5.4 Complejidad de los algoritmos 49

5.5 Conclusiones de las pruebas realizadas 51

Capítulo 6. Conclusiones y Trabajo Futuro 52

6.1 Líneas de Trabajo Futuro 52

Capítulo 7. Bibliografía 54

Capítulo 8. Anexo 55

8.1 Código Implementado 55

8.1.1 Simulación BER 55

8.1.2 BoxDetection_ADMIN 57

8.1.3 BoxDetection_OCD 58

8.1.4 Projeccion 59

8.1.5 sorMethod 59

8.1.6 SLAS 60

8.1.7 calc_NL 63

Capítulo 1. Introducción, Motivación y Objetivos

Desde hace más de 5 años se ha podido observar un aumento considerable en el uso de vehículos aéreos no tripulados (UAV) [1], comúnmente conocidos como drones, tanto para uso recreativo como para uso profesional y militar. Estos vehículos aéreos suponen una industria que no ha parado de crecer y que ha generado grandes beneficios en los últimos años. En la Figura 1 se puede ver dicho crecimiento, en cada uno de los sectores mencionados, a nivel mundial.

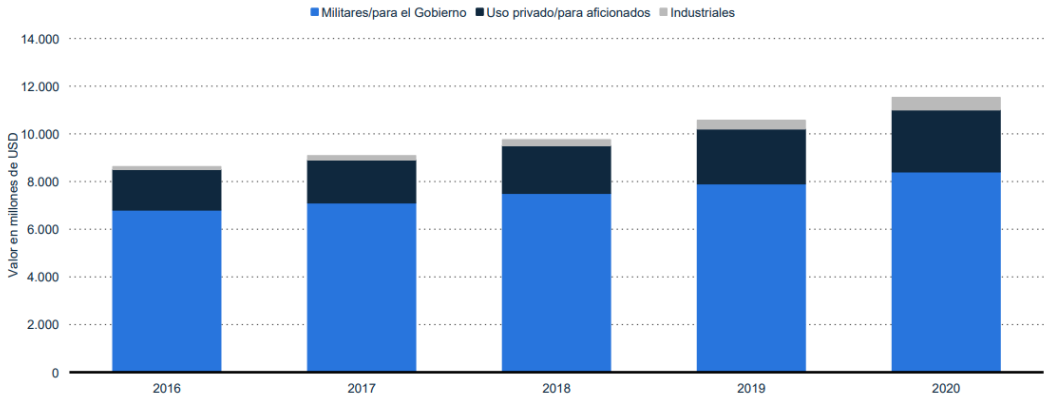


Figura 1: Valor de los segmentos de mercado de la industria de vehículos aéreos no tripulados a nivel mundial.

El sector comercial en particular a experimentado un gran crecimiento en la facturación y se espera que siga aumentando, al menos durante los próximos cuatro años, en la Figura 2 se puede observar este crecimiento. Esta popularidad se debe a la versatilidad que presentan los drones para realizar una gran variedad de tareas que, de realizarse de forma tradicional, suponen un gasto importante de tiempo y/o dinero.

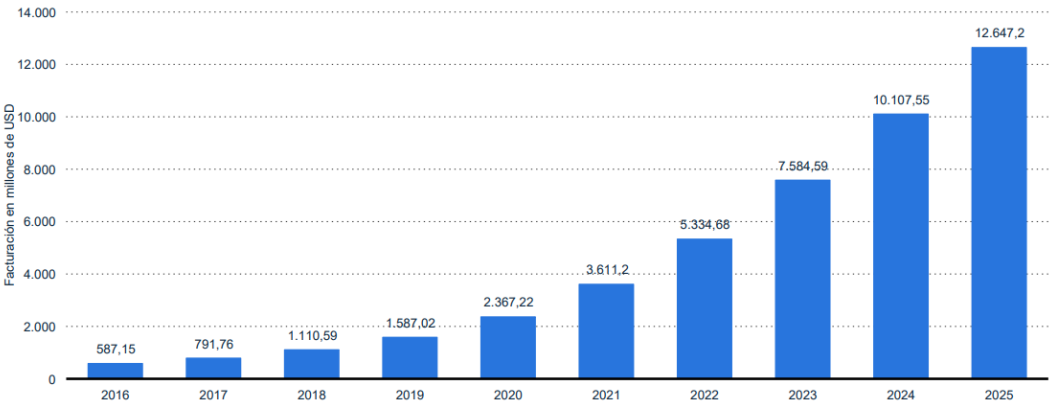


Figura 2: Facturación de la industria de drones comerciales a nivel mundial de 2016 a 2025.

Los usos comerciales son muy diversos y engloban diferentes industrias. En la Figura 3 se puede ver que el sector con más crecimiento previsto para 2020 es el de la filmación, con una gran diferencia sobre el resto. La introducción de UAV en estos sectores ha permitido realizar las tareas de una forma más eficiente y, además, realizar otras que hasta ese momento eran imposibles. Por ejemplo, en el caso de la filmación las tomas aéreas y en el sector agrícola la gestión de los cultivos o el mapeado del terreno.

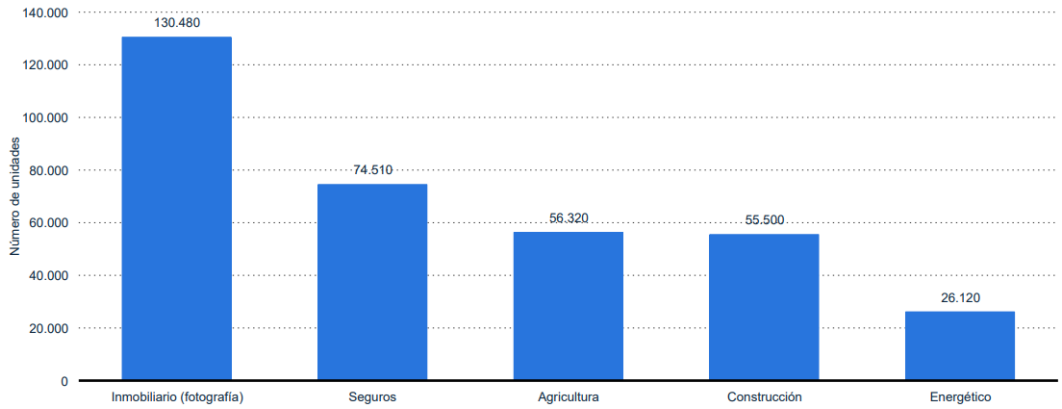


Figura 3: número de vehículos aéreos no tripulados comerciales utilizados en distintos sectores a nivel mundial en 2020.

Aunque la pandemia ha tenido un gran impacto en todos los sectores, en el caso de los drones este no ha disminuido su crecimiento, sino todo lo contrario, ha aumentado. Esto es debido a que la visión que se tiene de la inclusión de los drones en los diferentes sectores es con el objetivo de reducir las tareas realizadas por los humanos, automatizando los procesos. Esto permite que un operador humano pueda estar realizando las tareas desde cualquier lugar, sin necesidad de desplazarse.

El mayor exponente de esta aceleración del sector de drones es el reparto de paquetes a domicilio. Esta tarea es de especial interés y en la que se lleva trabajando mucho tiempo en búsqueda de una solución. En 2013 fue anunciado por Amazon el servicio de paquetería aérea *Prime Air*, el cual entregaría a los clientes sus paquetes utilizando drones. Desde entonces se han ido presentando nuevas patentes para llevar más allá este planteamiento, con el objetivo de crear un sistema autónomo de reparto con drones. La Figura 4 corresponde a dos patentes presentadas por Amazon, la primera representa un concepto para que los drones puedan identificar los gestos realizados por los clientes [2] (indicaciones para el aterrizaje) y la segunda describe lo que correspondería a un almacén aéreo, desde el cual los drones entregarían los paquetes a los clientes y después serían transportados de vuelta a dicho almacén [3].

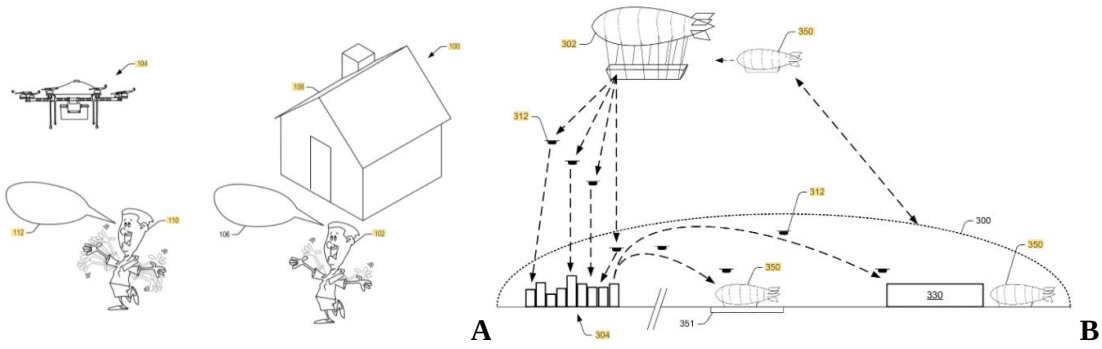


Figura 4: Patente de interpretación de gestos humanos (A). Patente de sistema de almacén aéreo (B).

En el caso de España, se ha desarrollado un plan estratégico en los últimos 3 años donde se espera que el número de UAV crezca a un ritmo acelerado hasta 2035 [4], a partir de este año la curva de crecimiento será moderada hasta el 2050. Para que este crecimiento sea posible se están llevando a cabo revisiones a la normativa existente para crear un marco, acorde con normativas europeas, que permita el desarrollo profesional de las actividades. Se espera que para 2035 el volumen de negocio llegue a los 1.220 M€ y cree 11.000 nuevos puestos de trabajo. En 2050 el sector seguirá aumentando y llegará a un volumen de 1.520 M€ y 11.500 puestos de trabajo. Sin embargo, este plan fue desarrollado años antes de la situación de pandemia que se ha vivido. Por lo tanto, es de esperar que el volumen de negocio y los puestos de trabajo creados sean superiores a las estimaciones realizadas.

España es el cuarto productor agrícola de la Unión Europea, por lo que se espera que una parte importante del volumen de negocio se concentre en este sector. Según el informe, el uso de drones se centrará en dos actividades: la toma de datos y el transporte y pulverización de productos para el cuidado de los cultivos. En la Figura 5 se puede ver la flota de drones a la que se espera llegar y el volumen de negocio en 2035 y 2050 comparado con la Unión Europea.

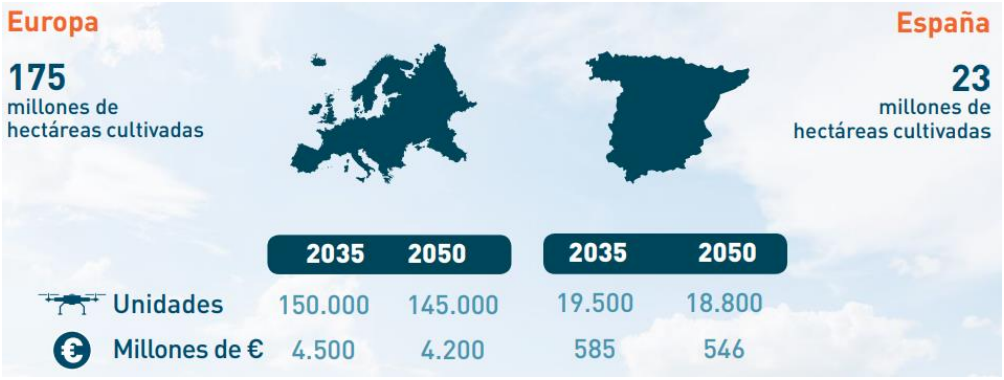


Figura 5: Previsión de flota y volumen de negocio en 2035 y 2050.

Un factor muy importante para que este rápido crecimiento ocurra es la exposición del sector al gran público. Esto se conseguirá con tareas visuales como construcción, comercio electrónico y paquetería y entretenimiento, entre otras. Es por esto que las carreras de drones juegan un papel importante para la divulgación, ya que se presentan como una forma sencilla de atraer el interés del público.

Las carreras se realizan con distintos tipos de drones, aunque las que más éxito han tenido en todo el mundo son las de drones con 4 rotores de envergadura mini (según las denominaciones del plan estratégico). Durante el resto del artículo la palabra drones se referirá únicamente a aquellos con 4 rotores.

Estas carreras comenzaron como eventos que se organizaban entre grupos de aficionados, no fue hasta el 2015 cuando empezaron a aparecer diferentes organizaciones que crearon ligas para realizar competiciones alrededor del mundo. En los últimos años han ganado popularidad, propiciando que sigan apareciendo más ligas en diferentes países. En el caso de España, en 2019 se fundó la IDL (*Iberian Dron League*), una competición que organiza carreras en diferentes ciudades de España tanto en espacios abiertos como cerrados. A nivel mundial, las organizaciones más conocidas son la *DRL* (*Dron Racing League*) y *DCL* (*Dron Champions League*) ya que son de las pocas que retransmiten por televisión sus eventos. En la figura 6 se puede ver el circuito que se creó en el gran premio de 2019 de IDL en la ciudad de Valencia.

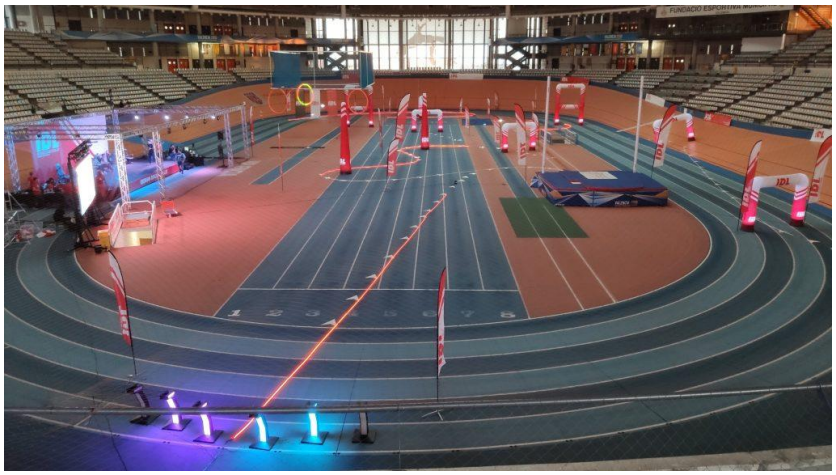
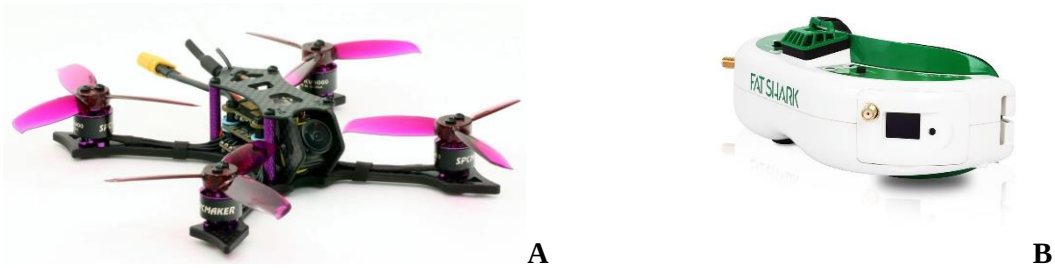


Figura 6: Circuito de Valencia de IDL en 2019.

Las carreras ocurren en una jornada y constan de cuatro fases diferentes. El circuito es el mismo en todas las fases, por lo que no se produce ningún desplazamiento durante toda la jornada.

Los drones están equipados con cámaras en la parte delantera para que el piloto, mediante unas gafas FPV (*First Person View*), pueda tener el punto de vista del dron y ver en todo momento los obstáculos que pueda tener delante, aun sin NLOS (*Non-Line Of Sight*). En la figura 7 se puede ver un ejemplo de un dron de carreras y unas gafas FPV. Para transmitir la señal se utiliza un VTX (*Video Transmitter*) a una frecuencia de 5.8 GHz, este es independiente del utilizado para recibir la señal de control en el dron. Las limitaciones que sufren es la limitación del número máximo de participantes en una carrera (8, en algunos países menos) y la limitación de potencia máxima que puede transmitir el dron, esto se debe a que una mayor potencia podría causar interferencias entre los VTX de los pilotos.



A **B**

Figura 7: Imagen de dron de carreras (A) y gafas FPV (B).

Los sistemas MIMO (*Multiple-Input Multiple-Output*) utilizan múltiples antenas en el transmisor y receptor con el objetivo de aumentar la eficiencia espectral, el alcance o la fiabilidad del enlace. La tecnología MIMO no es una novedad en la actualidad, lleva presente en el estándar de telefonía móvil del 3GPP [5] desde la tercera generación (3G). MIMO se ha explotado en la 4G empleando un número de antenas contenido y con la idea de que los sistemas mantuvieran una relación de antenas transmisor receptor cercana a 1 (2x2, 4x4, 8x4, etc.). MIMO hace uso de técnicas como diversidad, multiplexación espacial y beamforming para ofrecer estas mejoras.

La diversidad puede estar presente tanto en el receptor como en el transmisor y consiste en el uso de múltiples antenas para la transmisión o recepción de una señal. En recepción, se puede actuar de dos formas, seleccionando una de las múltiples señales recibidas y, en función de ciertos criterios de la señal, elegir la considerada como más fiable o mediante una combinación ponderada de las señales recibidas. Por otro lado, en transmisión, la técnica más utilizada es la codificación STC (*Space-Time Coding*). Esta técnica consiste en transmitir múltiples copias de datos sobre múltiples antenas de forma que el receptor pueda extraer la mayor información posible y que los datos obtenidos sean fiables.

En multiplexación espacial se hace uso de varias antenas para transmitir una señal con una alta eficiencia (b/s/hz) mediante la transmisión de señales independientes con una eficiencia menor. Para poder emplear multiplexación espacial en un sistema MIMO este ha de tener un número de antenas receptoras igual o superior al de antenas transmisoras. Esta técnica también puede usarse para transmitir simultáneamente a diferentes usuarios.

Por último, beamforming permite controlar el patrón de radiación de la antena y así aumentar la potencia de la señal en una determinada dirección. Esta técnica es muy útil en espacios concurridos con una gran cantidad de sistemas de comunicación diferentes, ya que actúa como un filtro espacial reduciendo las interferencias al concentrar la potencia radiada en una zona concreta.

Massive MIMO (mMIMO) se presenta en 5G como una continuación de las técnicas MIMO empleando un número de antenas de un orden superior respecto a las utilizadas hasta ahora en MIMO. Esto hace que una BS (*Base Station*) pueda alcanzar un rendimiento mayor, permitiendo que pueda dar servicio a más usuarios de forma simultánea. De esta manera, mMIMO es una opción muy atractiva como respuesta a la demanda de velocidad y fiabilidad de la creciente cantidad de dispositivos conectados (drones, vehículos inteligentes, dispositivos IoT, etc).

Como objetivo de este trabajo se quiere explorar el uso de 4 algoritmos de detección de mMIMO para conocer con cual se obtiene un mejor resultado empleando una modulación OFDMA (*Orthogonal Frequency Division Multiple Access*) para la transmisión de video entre los drones y la BS. De esta manera, se pretende demostrar si mediante una BS de mMIMO es posible mejorar la calidad del video transmitido y de la reproducción del mismo como contenido bajo demanda, facilitando así la difusión de este deporte.

En este trabajo se propone el despliegue de un sistema de comunicación de una BS mMIMO que actúe como receptora de la señal transmitida por los drones y la distribuya a los pilotos con el objetivo de mejorar las prestaciones de los sistemas utilizados en la actualidad.

En la Figura 8 se puede observar un esquema de cómo sería la comunicación. Los drones en el aire transmitirían la señal de video a la BS y esta será la encargada de retransmitirla a las gafas de los pilotos, ya sea a través de una conexión por cable o inalámbrica. Además de dar servicio a los pilotos la BS también se comunicaría con el servidor encargado de ofrecer el contenido a los usuarios.

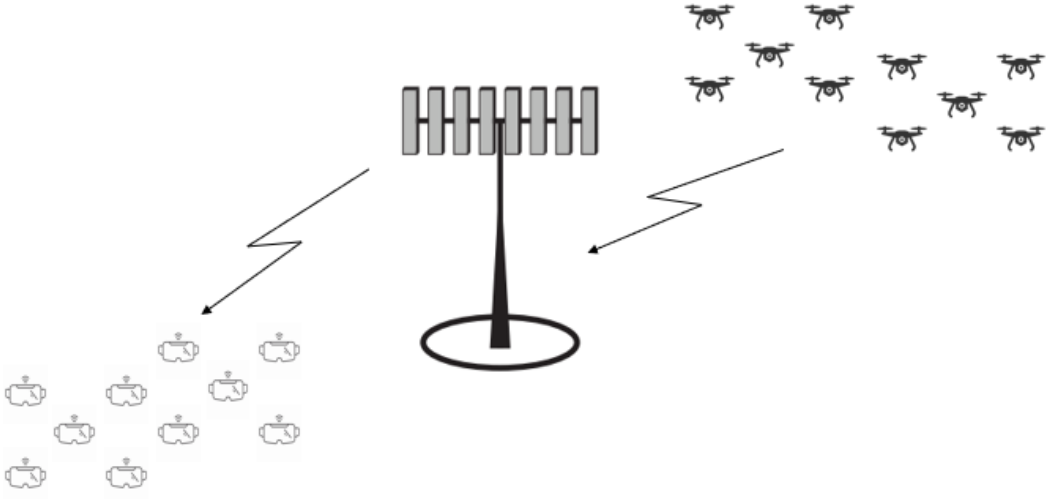


Figura 8: Despliegue de BS propuesto para carreras de drones.

Capítulo 2. Estado del arte

Las carreras de drones de las principales competiciones tienen lugar en circuitos montados dentro de estadios o pabellones deportivos que se habilitan para esta ocasión. Los sistemas de comunicación que se utilizan son comerciales y no incluyen ninguna característica específica, algunas competiciones tienen sus propios drones y no permiten el uso de drones propios pero las características de los sistemas VTX son las mismas. Se utilizan dos sistemas separados para la comunicación, uno para la comunicación entre el dron y la controladora de vuelo que utiliza el piloto para controlar el dron y otro para la transmisión de video a las gafas FPV.

2.1 VTX Analógico y Digital

Los VTX analógicos funcionan a una frecuencia de 5.8 GHz y con potencias que pueden ir desde los 25 mW a los 600 mW. La potencia que se emplea en las carreras es de 25 mW debido a que los drones vuelan unos cerca de los otros y en espacios cerrados. En sistemas analógicos existen 10 bandas con 8 canales por banda. En la Figura 9 se puede ver la distribución de las típicas bandas y canales en frecuencia que son utilizados. La banda R (*Racing*) suele utilizarse para las carreras y es la que dispone de una mayor separación entre canales, 37 MHz.

Band	CH 1	CH 2	CH 3	CH 4	CH 5	CH 6	CH 7	CH 8
A	5865	5845	5825	5805	5785	5765	5745	5725
B	5733	5752	5771	5790	5809	5828	5847	5866
E	5705	5685	5665	5645	5885	5905	5925	5945
F	5740	5760	5780	5800	5820	5840	5860	5880
R	5658	5695	5732	5769	5806	5843	5880	5917
D	5362	5399	5436	5473	5510	5547	5584	5621
U	5325	5348	5366	5384	5402	5420	5438	5456
O	5474	5492	5510	5528	5546	5564	5582	5600
L	5333	5373	5413	5453	5493	5533	5573	5613
H	5653	5693	5733	5773	5813	5853	5893	5933

Figura 9: Distribución en frecuencia de las bandas y canales.

Los canales se han de elegir con cuidado para evitar interferencias y, debido a esta limitación, el número de drones que pueden volar de forma simultánea no supera los 8 participantes. Esta es la razón que imposibilita que se organicen carreras con un mayor número de pilotos.

El sistema VTX que se encuentra en el dron se puede dividir en tres partes: cámara, VTX y antena. La cámara empleada en drones de carreras no supera el tamaño de 20 mm de grosor y solo tiene 3 conexiones: entrada de alimentación, toma a tierra y salida de vídeo. El formato de video transmitido es PAL o NTSC (ambos con relación de aspecto 4:3), lo que ofrece una resolución máxima de 720x576 y 720x480.

El módulo VTX es el encargado de recibir la señal analógica de vídeo generada por la cámara y transmitirla mediante una señal analógica. Se pueden encontrar diferentes tipos de VTX en el mercado que, según su rango de precio, incluirá más o menos conexiones en función de las funcionalidades que ofrezca. Como muestra de ejemplo, en la Figura 10 se ha escogido un VTX con 5 conexiones: alimentación, video, salida de voltaje y 2 salidas a tierra, además del conector RP-SMA de la antena.

Figura 10: Módulo VTX.

La entrada de alimentación es de un voltaje de entre 7v y 24v, este valor es estándar para la gran mayoría de VTX que se encuentran en el mercado, y da potencia al módulo. La entrada de video es el puerto por el que se recibe la señal analógica generada por la cámara y que se transmite al receptor en las gafas FPV. Normalmente se incluyen dos salidas de tierra, aunque es la misma dividida en dos.

Dependiendo del tamaño y precio del módulo este puede incluir una salida de voltaje pensada para alimentar la cámara de video. La ventaja de utilizar esta salida es que se realiza un filtrado que limpia la señal que llega de la batería, resultando en una señal de vídeo más nítida que si se alimenta directamente desde la batería del dron. Por último, también se puede encontrar un conector dedicado a Smart audio/Tramp, un protocolo que comunica la controladora de vuelo del piloto con el VTX para modificar parámetros de visionado del video en las gafas FPV.

Los sistemas digitales tienen unas características físicas similares a los analógicos y están compuestos por las mismas partes (cámara, transmisor y antena). Al ser un producto novedoso en el mercado no existen un gran número de marcas que los fabriquen. Para este trabajo se ha tomado como referencia las especificaciones de los productos de 2 marcas, DJI y Caddx, ya que son de las más conocidas a la hora de comprar estos productos.

Las cámaras digitales tienen una resolución de 720p (HD) y el resto de características son iguales a las de una cámara analógica. Las cámaras digitales no permiten conectar los cables de alimentación directamente a la batería, todos los cables están conectados al módulo VTX. El tamaño es el mismo que en cámaras analógicas para facilitar la compatibilidad a la hora de montarlas en un dron.

El módulo VTX digital sufre del mismo problema que su contraparte analógica, interferencias cuando el número de pilotos es elevado. En el caso de la marca DJI, el VTX dispone de 8 canales diferentes, lo que permite que 8 drones vuelen simultáneamente, aunque la cantidad de canales disponibles variará en función de la región (FCC 8 y CE 4). Todos los canales funcionan con una frecuencia central de 5.8 GHz y están separados entre sí 20 MHz.

A diferencia de los VTX analógicos, las versiones digitales tienen una mayor latencia. Esto se debe principalmente al preprocesado digital que realiza, ya que esta conversión analógico-digital no se encuentra presente en las versiones analógicas. Sin embargo, al comparar la calidad del video analógico con el digital se observa a simple vista que la imagen digital es menos borrosa y representa mucho mejor los colores que su contraparte analógica. Además, la resolución de la imagen es mayor y se puede alcanzar una tasa de fotogramas por segundo de 120. La latencia del módulo de DJI varía en torno a 28 ms (720p, 120fps) o 40 ms (720p, 60fps).

2.2 Gafas FPV

Al igual que los transmisores, también se encuentran versiones analógicas y digitales de las gafas utilizadas por los pilotos. Actualmente en el mercado, la compañía FatShark [6] es la más popular y la que ofrece una gran variedad de modelos de gafas analógicas. En cuanto a versiones digitales, la compañía DJI es la que ofrece unas gafas con las mejores características [7]. En la Figura 11 se pueden ver modelos recientes de ambas compañías.

A **B**

Figura 11: DJI FPV (A) y FatShark Attitude V6 (B).

En cuanto a resolución, se pueden encontrar versiones analógicas que superan los 1440x810 píxeles en pantalla y los 30º a 54º de campo de visión de las gafas de DJI. Sin embargo, estas gafas digitales son las que presentan una mayor tasa de refresco, 144 Hz.

La principal ventaja de las versiones analógicas es que su tecnología es madura y existen una gran variedad de modelos con diferentes especificaciones. Aun siendo una tecnología con un largo recorrido, las versiones analógicas no se han quedado atrás, estas incluyen funcionalidades para dar soporte a la reproducción de vídeo digital incluyendo un puerto HDMI, haciendo que puedan tener más usos.

La compañía Fatshark ha presentado una nueva propuesta para adaptar la tecnología digital a sus gafas FPV analógicas. En la Figura 12-A se puede ver el receptor digital externo que se adhiere a las gafas, habilitando la reproducción de video digital con una resolución 1280x720 a 60 fotogramas por segundo. Esta solución, aunque utiliza unas gafas analógicas, requiere de una cámara y transmisor digital. En la Figura 12-B se muestra el resultado final una vez montado el módulo.

A **B**

Figura 12: Módulo digital SharkByte (A), Módulo montado en gafas analógicas (B).

2.3 Antenas utilizadas en transmisión y recepción

Existen diferentes tipos de antenas tanto para el VTX que se encuentra en el dron como para las gafas FPV. En el dron, el VTX permite la conexión de una antena, mientras que las gafas FPV permiten la conexión de 2 o más antenas. En ambos casos el conector utilizado es RP-SMA o SMA.

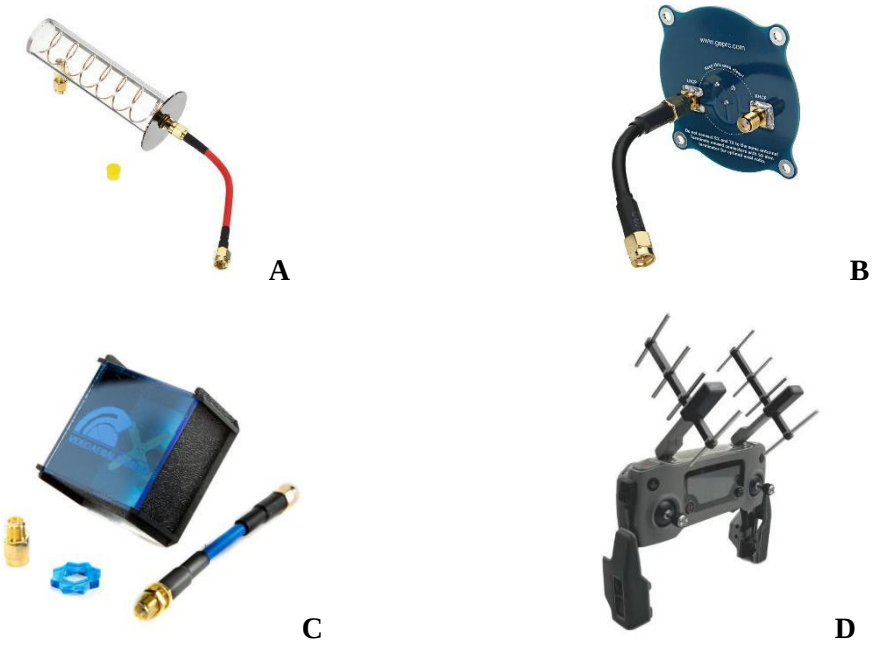
Dependiendo del uso que se haga con el dron unas antenas serán mejor opción que otras. En el caso de una carrera se conoce el espacio en el que volará el dron, por lo tanto, se podrán utilizar antenas con mayor directividad ya que tienen una mayor ganancia. En cambio, una antena directiva no servirá si el dron está volando a nuestro alrededor ya que habría puntos en los que la antena no tendría visión directa. En este caso, una mejor opción sería una antena omnidireccional.

Ya que los drones se mueven en el espacio en 3 dimensiones, la antena transmisora rotará y difícilmente se encontrará en la misma posición. Es por esto que las antenas más populares para FPV son con polarización circular. De esta forma se asegura una buena recepción de la señal independientemente del ángulo de rotación del dron respecto a la posición del piloto. Es por esto que las antenas más utilizadas para FPV son las omnidireccionales con polarización circular. En la Tabla 1 se pueden ver los 6 tipos diferentes de antenas que se pueden encontrar.

	Polarización	**Radiación**
Helicoidal	Circular	Direccional
Parche	Circular o lineal	Direccional
Crosshair	Circular	Direccional
Yagi	Lineal	Direccional
Clover Leaf	Circular	Omnidireccional
Skew Planar	Circular	Omnidireccional
Pagoda	Circular	Omnidireccional

Tabla 1: Clasificación de antenas utilizadas en transmisión y recepción.

Las antenas Helicoidales, Parche, Crosshair y Yagi se utilizan para aumentar el alcance en una dirección concreta. Son útiles en vuelos largos y, en algunos casos, se utilizan junto a una estación de seguimiento para mantener las antenas alineadas. Debido a su direccionalidad, no se utilizan como antenas transmisoras. En la Figura 13 se puede ver la forma de estas antenas.



A **B** **C** **D**

Figura 13: Antenas Helicoidal (A), Parche (B), Crosshair (C) y Yagi (D).

Las antenas Clover Leaf y Skew Planar se diferencian en el número de "hojas" que tiene la antena. Como se puede ver en la Figura 13, Clover Leaf tiene 3 hojas y Skew Planar 4. Estas antenas son muy utilizadas por su bajo precio y sirven en la gran mayoría de condiciones.

A **B**

Figura 13: Comparación antenas Skew Planar (A) y Clover Leaf (B).

La antena pagoda es relativamente nueva, apareció en 2016, y se planteó como una mejora respecto al resto de antenas omnidireccionales. Las láminas circulares que se observan en la Figura 14 están hechas del mismo material que las placas de circuito impreso (PCB) facilitando su fabricación.

Figura 14: Antena Pagoda.

Para mejorar la recepción de la señal, una técnica que se ha adoptado en los visores FPV es la posibilidad de conectar dos antenas para aprovechar la diversidad. De esta manera se pueden combinar los beneficios de las antenas omnidireccionales y las direccionales para adaptarse al contexto en el que se esté volando el dron. Por ejemplo, en una carrera se podrían combinar dos antenas direccionales para cubrir todo el circuito.

2.4 Ventajas y desventajas introducidas por mMIMO

Para introducir mMIMO en el contexto de las carreras de drones se plantea el uso de una BS mMIMO que sea la que reciba la señal de video de los drones y la retransmita a las gafas FPV de los pilotos. De esta manera será la BS la encargada de recuperar la señal de video (detección, demodulación, etc.), teniendo esta una mayor capacidad de procesado que unas gafas FPV. Este trabajo se centra en la comunicación entre la BS y drones, la comunicación entre la BS y las gafas FPV puede realizarse de diferentes maneras debido a que los pilotos se encuentran en un punto fijo durante toda la carrera. La manera más sencilla y fiable es comunicarlos utilizando una conexión por cable, como se muestra en la Figura 15.

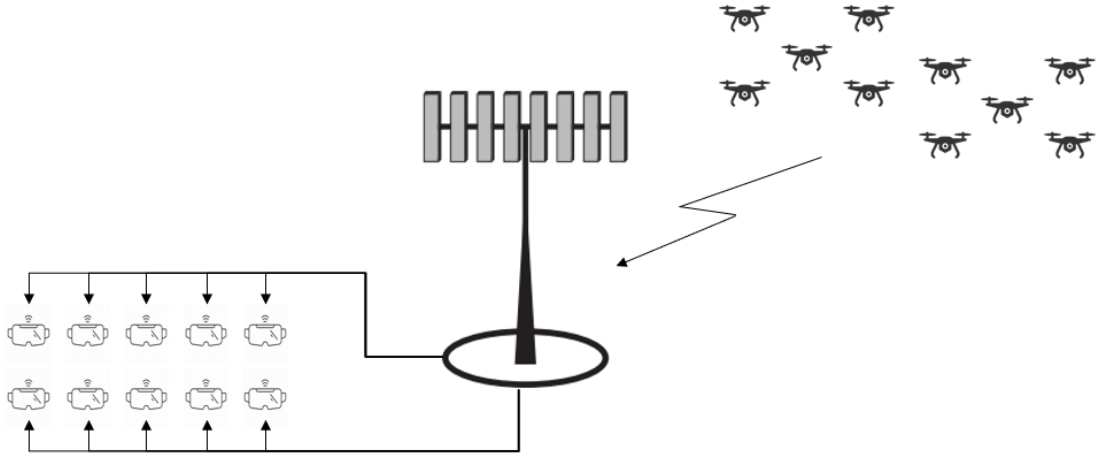


Figura 15: Despliegue mMIMO propuesto para carreras de drones.

La introducción de esta tecnología beneficiará tanto a los pilotos como a los espectadores, mejorando la calidad del servicio (QoS) y la calidad de experiencia (QoE). Las ventajas aportadas por mMIMO son varias.

La comunicación con la BS seguiría el estándar 5G, por lo tanto, se utilizaría la modulación OFDMA. De esta manera se realiza un uso más eficiente del espectro, permitiendo un **mayor número de transmisiones simultáneas**. Esto es posible debido a que el número de antenas receptoras es mucho mayor que el número de antenas transmisoras y es posible realizar multiplexación espacial.

En mMIMO, al tener múltiples antenas en el receptor es posible recuperar la señal transmitida por cada dron por parte de la BS (se supone CSI). Esto permite que cada transmisión se realice empleando todo el ancho de banda y, así, **aumentando la capacidad de cada canal**. Esta mejora permitirá transmitir contenido de mayor resolución y/o de mayor tasa de fotogramas por segundo, incrementando la QoE tanto de los pilotos como de los espectadores.

Otra ventaja de utilizar mMIMO es el **aumento del radio de cobertura** de la señal sin necesidad de aumentar la potencia transmitida. Esto se consigue por la elevada cantidad de antenas receptoras en la BS, ya que si para un valor de SNR dado se obtiene un BER (*Bit Error Rate*) inferior es posible aumentar el alcance.

La cobertura que se tendrá con la BS será mayor que la obtenida con la antena receptora de las gafas FPV, permitiendo que con una BS sea suficiente para cubrir la totalidad de los circuitos y dará la oportunidad de ampliarlos. La ventaja de aumentar el rango de cobertura de esta forma es evitar las posibles restricciones de potencia de las VTX de los drones en las diferentes regiones.

Por último, utilizar una BS para transmitir la señal de video **mejorará la latencia** de la señal debido a que esta es capaz de procesar una mayor cantidad de datos que las propias gafas FPV. Es por esto que bajo las mismas condiciones (transmitiendo a un mismo bitrate) la BS reducirá la latencia, mejorando la experiencia del piloto.

Por otro lado, hay 2 desventajas al desplegar una BS para cada uno de estos eventos. Por un lado, el **coste de una BS es elevado,** haciendo que las organizaciones sean reticentes a asumir el coste y, por otro lado, debido a las dimensiones de una BS, el **despliegue puede ser aparatoso**.

Estas desventajas pueden ser un impedimento al principio y pueden provocar que disminuya el interés de incluirla en las carreras. Sin embargo, una vez se haga uso de BS las ventajas superarán las desventajas. Con el tiempo, el coste de las BS se amortizará y, si se sigue fomentando el crecimiento de este deporte, su despliegue ya no supondrá un coste adicional.

Capítulo 3. Evaluación de los Costes del Proyecto

En este apartado se van a exponer los costes temporales y económicos que se han planificado para el desarrollo de este proyecto. Para ello se han tenido en cuenta las horas de trabajo dedicadas a cada tarea y el coste del material utilizado, ya sea hardware o software.

3.1 Costes económicos del proyecto

La realización de este proyecto no ha requerido de la adquisición de hardware nuevo para realizar las diferentes tareas del proyecto. Las licencias software que se han utilizado no han supuesto ningún coste adicional ya que todas se encuentran dentro de las licencias educativas proporcionadas por la Universidad Politécnica de Valencia.

Para las tareas de planificación del proyecto se ha hecho uso de la aplicación Project de Microsoft Office, con la que han creado las diferentes tareas y se ha podido realizar una gestión de los recursos y planificación temporal de las mismas.

Los artículos leídos de los diferentes algoritmos de detección utilizados en mMIMO se han accedido mediante el portal de IEEE Explorer [8], donde se ha accedido con la licencia de la institución de la Universidad Politécnica de Valencia.

Como lenguaje de programación para la implementación de las diferentes funciones se ha elegido Matlab. La ventaja que ofrece sobre otros lenguajes de programación es el uso de módulos (librerías) de comunicaciones móviles que incluyen funciones para modular, demodular y generar diferentes tipos de canales.

Para la aplicación de diferentes filtros de video se ha hecho uso de la herramienta FFmpeg [9]. En concreto, se ha utilizado para modificar parámetros de los videos como el bitrate, fotogramas por segundo o la resolución. También se ha utilizado para comprobar la calidad del video recibido mediante la puntuación VMAF.

Durante el desarrollo del proyecto se ha hecho uso de diferentes máquinas para la ejecución de las simulaciones. Por un lado, se ha utilizado un ordenador de sobremesa con las siguientes especificaciones:

- CPU: Intel core i7-6700k 4 GHz.
- GPU: Nvidia GeForce GTX 1070.
- RAM: DDR4 2400 PC4-19200 4x8GB.

Esta máquina ha sido aportada por el alumno, por lo tanto, no ha supuesto ningún coste para el proyecto. Por otro lado, durante el periodo en el que se ha realizado el proyecto, se han realizado prácticas extracurriculares en el Instituto de Telecomunicaciones y Aplicaciones Multimedia (iTEAM), dentro del Grupo de Tratamiento de Audio y Comunicaciones (GTAC). Durante la realización de las prácticas se ha tenido acceso a un servidor en el que se han ejecutado las simulaciones más costosas. El servidor cuenta con las siguientes especificaciones:

- CPU: Intel Xenon 5680.
- GPU: Nvidia GTX 960 x2.
- RAM: DDR2 96GB.
- SO: Ubuntu 16.04.2 LTS.

El uso del servidor ha sido de especial utilidad para poder ejecutar diversas simulaciones de forma simultánea que requieren de varios días para completarse. Por otro lado, las matrices de canal son de grandes dimensiones debido a la gran cantidad de antenas en el receptor, haciendo que en memoria ocupen mucho espacio. Al tener el servidor una gran cantidad de memoria RAM, ha permitido ejecutar simulaciones que, de haberse hecho en el ordenador de sobre mesa, no habrían sido posibles.

Según lo comentado anteriormente, el desarrollo del proyecto no ha supuesto ningún coste económico ya que no ha sido necesario adquirir ningún componente hardware o software, todos los utilizados ya se poseían con antelación.

3.2 Costes temporales del proyecto

A continuación, se van a desglosar las diferentes tareas realizadas durante el proyecto y el tiempo dedicado a cada una de ellas. En la Figura 16 se puede observar un EDT (Estructura de Descomposición de Tareas) de las tareas del proyecto. Como se puede observar en la figura, el proyecto ha quedado dividido en 5 subsistemas. En el primero se plantea la estructura del proyecto definiendo que tareas se van a realizar y en qué orden. En este punto también se define el alcance del proyecto.

En el segundo subsistema se realizan las tareas de búsqueda de información para conocer el estado del arte de las técnicas de mMIMO y de la transmisión de video en carreras de drones. El objetivo de este apartado es conocer que tecnologías se utilizan en las carreras de drones y cual sería la mejor manera de introducir la tecnología mMIMO para su uso en las carreras. Por otro lado, también se realiza una búsqueda de diferentes algoritmos de detección empleados para elegir los óptimos para el contexto de una carrera de drones.

En el tercer subsistema se realiza la implementación de los detectores elegidos en Matlab y de un canal mMIMO en el que poder realizar transmisiones de datos con el objetivo de evaluar el rendimiento de los detectores.

En el cuarto subsistema se realizan las pruebas requeridas utilizando los algoritmos implementados en Matlab previamente. El objetivo es validar la utilidad de los detectores escogidos y realizar una transmisión de un video en las mejores condiciones de cada detector. Por último, en el quinto subsistema, se realiza la redacción de la memoria del proyecto. Durante la redacción de la memoria se mantienen reuniones periódicas con el tutor.

1.	**Proyecto TFM**
1.1	**Planificación del proyecto**
1.1.1	**Objetivo:** Definir la estructura del proyecto y las tareas a realizar
1.1.2	**Actores:** Tutor TFM y Alumno
1.1.3	Definición de la estructura del proyecto
1.1.4	Definición de las tareas a realizar
1.1.5	Reunión con el tutor
1.2	**Investigación del estado del arte**
1.2.1	**Objetivo:** Conocer el estado del arte de mMIMO y de la transmisión de video en carreras de drones
1.2.2	**Actores:** Tutor TFM y Alumno
1.2.3	Técnicas mMIMO
1.2.4	Algoritmos de detección empleados en mMIMO
1.2.5	Técnicas de transmisión de video en carreras de drones
1.2.6	Selección de algoritmos óptimos para carreras de drones
1.2.7	Reunión con tutor
1.3	**Creación del banco de pruebas**
1.3.1	**Objetivo:** Desarrollar un entorno software para realizar pruebas con los algoritmos seleccionados
1.3.2	**Actores:** Tutor TFM y Alumno
1.3.3	Implementación de canal mMIMO en Matlab
1.3.4	Validación del canal mMIMO implementado
1.3.5	Implementación de funciones de algoritmos óptimos seleccionados
1.3.6	Validación de las funciones implementadas
1.3.7	Reunión con tutor
1.4	**Realización de pruebas**
1.4.1	**Objetivo:** Validar el uso de los algoritmos seleccionados en carreras de drones
1.4.2	**Actores:** Tutor TFM y Alumno
1.4.3	Simulaciones para la obtención del BER
1.4.4	Simulación de transmisión de video
1.4.5	Evaluación de resultados de las simulaciones
1.4.6	Reunión con tutor
1.5	**Memoria TFM**
1.5.1	**Objetivo:** Redactar la memoria del trabajo final de máster
1.5.2	**Actores:** Tutor TFM y Alumno
1.5.3	Redacción de la memoria

Figura 16: EDT del Trabajo Final de Máster.

La mayor parte del tiempo dedicado a este proyecto se encuentra en las pruebas realizadas para el cálculo del BER y de la transmisión de un video. Debido al gran número de antenas las multiplicaciones de la matriz de canal y de datos tardan más tiempo en realizarse, lo que hace que las simulaciones tomen mucho tiempo. En la Figura 17 se muestra el tiempo asignado a cada tarea del proyecto, siendo el apartado de realización de pruebas y la realización de la memoria los apartados que más tiempo requieren.

	Fecha de inicio	Fecha fin	Duración (días)
Proyecto TFM	10/02/2021	20/08/2021	138
Planificación del proyecto	10/02/2021	19/02/2021	8
Definición de la estructura del proyecto	10/02/2021	12/02/2021	3
Definición de las tareas a realizar	15/02/2021	18/02/2021	4
Reunión con el tutor	19/02/2021	19/02/2021	1
Investigación del estado del arte	22/02/2021	22/03/2021	21
Técnicas mMIMO	22/02/2021	19/03/2021	20
Algoritmos de detección empleados en mMIMO	22/02/2021	12/03/2021	15
Técnicas de transmisión de video en carreras de drones	22/02/2021	05/03/2021	10
Selección de algoritmos óptimos para carreras de drones	15/03/2021	19/03/2021	5
Reunión con tutor	22/03/2021	22/03/2021	1
Creación del banco de pruebas	23/03/2021	29/04/2021	28
Implementación de canal mMIMO en Matlab	23/03/2021	19/04/2021	20
Validación del canal mMIMO implementado	20/04/2021	21/04/2021	2
Implementación de funciones de algoritmos óptimos seleccionados	23/03/2021	26/04/2021	25
Validación de las funciones implementadas	27/04/2021	28/04/2021	2
Reunión con tutor	29/04/2021	29/04/2021	1
Realización de pruebas	30/04/2021	02/07/2021	46
Simulaciones para la obtención del BER	30/04/2021	24/06/2021	40
Simulación de transmisión de video	30/04/2021	24/06/2021	40
Evaluación de resultados de las simulaciones	25/06/2021	01/07/2021	5
Reunión con tutor	02/07/2021	02/07/2021	1
Memoria TFM	19/04/2021	20/08/2021	90
Redacción de la memoria	19/04/2021	20/08/2021	90
Reunión con tutor	20/04/2021	10/08/2021	81

Figura 17: Asignación temporal de las tareas realizadas durante el proyecto.

Capítulo 4. Algoritmos de Detección Empleados

Para conocer que algoritmos serían los más adecuados en el contexto de una carrera de drones se ha realizado una búsqueda exhaustiva en la bibliografía para conocer cuales son los que ofrecen unos mejores resultados. Tras esta búsqueda, se han seleccionado 4 como los más indicados para este caso. Algunos de los artículos leídos son [10] y [11]

El artículo [11] presenta una comparación entre diferentes detectores que son utilizados en **mMIMO** agrupados por técnicas de detección: detectores lineales, basados en *local search*, *belief propagation*, *box detection*, *sparsity based* y *machine learning*. Al ser este un artículo reciente y que agrupa diversas técnicas de detección, los 4 algoritmos se han seleccionado de este artículo.

Como criterio de selección se han comparado las ventajas y desventajas de los diferentes algoritmos en el contexto de carreras de drones para conocer si afectan de forma positiva o negativa al sistema. Los criterios que se han tenido en cuenta son la latencia introducida en el sistema, la tolerancia a canales variantes en el tiempo, la complejidad computacional, el rendimiento según el ratio de antenas y si se requiere preparación o no.

Una carrera de drones es un evento planificado, por lo tanto, que se requiera una preparación previa de algún tipo no tendrá un impacto negativo de por sí en la celebración de la carrera. Por otro lado, una desventaja crítica es la latencia introducida por el detector. Debido a las altas velocidades a las que vuelan los drones, los pilotos tienen un tiempo de reacción muy pequeño, si a esto se le suma un aumento en la latencia de la imagen que recibe el piloto debido al detector empleado, esto puede provocar que manejar los drones sea una tarea casi imposible para los pilotos.

En las carreras de drones, aunque se duplique el número actual de corredores simultáneos, la relación entre antenas transmisoras y receptoras no será cercana a 1. Debido a esto, se han priorizado los algoritmos que no ven afectado su rendimiento cuando esta relación es baja o que su rendimiento es independiente de dicha relación. De igual manera, los algoritmos que presentan una baja complejidad computacional también han sido priorizados respecto a los que no, ya que esto permite que los requisitos del hardware empleado puedan ser inferiores, reduciendo así los costes de la BS empleada. Por último, debido a la alta movilidad de los drones en las carreras, se ha tenido en cuenta la tolerancia de los algoritmos a canales variantes en el tiempo.

Según los requisitos comentados, se han terminado por elegir 4 algoritmos: OCD-BOX (*Optimized Coordinate Descent-based BOX Equalization*) [12], ADMIN (*ADMM-based Infinity Norm method*) [13], *SOR* (*Successive Over Relaxation method*) [14] y *SLAS* (*Sequential Likelihood Ascent Search*) [16].

4.1 Modelo del Sistema

En los cuatro algoritmos elegidos se considera un enlace mMIMO ascendente con *K* usuarios y *N* antenas receptoras en la BS. En todos los casos, *N* será mucho mayor que *K*. La señal recibida en la BS queda representada por la siguiente expresión matemática

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}. \tag{1}$$

Donde $\boldsymbol{y} = [\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_N}]^{\boldsymbol{T}}$ es el vector de dimensión $(N, 1)$ que representa la señal recibida en la BS, el vector de señales transmitidas por los *K* usuarios de dimensión $(K, 1)$ es representado por $\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_K}]^{\boldsymbol{T}}$, la matriz del canal MIMO es representada por $\boldsymbol{H}$ con dimensiones (N, K) y el vector de ruido de dimensión $(N, 1)$ es representado por $\boldsymbol{n} = [\boldsymbol{n_1}, \boldsymbol{n_2}, \dots, \boldsymbol{n_N}]^{\boldsymbol{T}}$.

En dos de los algoritmos utilizados, el modelo del sistema utilizará valores reales en vez de complejos. De esta manera, la señal recibida queda definida de la siguiente forma

$$\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{H}}\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{n}}. \quad (2)$$

En este caso la señal recibida es representada por $\tilde{\boldsymbol{y}} = [\Re\{\boldsymbol{y}\} \quad \Im\{\boldsymbol{y}\}]^{\boldsymbol{T}}$, la señal transmitida es $\tilde{\boldsymbol{x}} = [\Re\{\boldsymbol{x}\} \quad \Im\{\boldsymbol{x}\}]^{\boldsymbol{T}}$, la matriz de canal es $\tilde{\boldsymbol{H}} = \begin{bmatrix} \Re\{\boldsymbol{H}\} & -\Im\{\boldsymbol{H}\} \\ \Im\{\boldsymbol{H}\} & \Re\{\boldsymbol{H}\} \end{bmatrix}$ y el vector de ruido es $\tilde{\boldsymbol{n}} = [\Re\{\boldsymbol{n}\} \quad \Im\{\boldsymbol{n}\}]^{\boldsymbol{T}}$. Al pasar a números reales el tamaño de las dimensiones se duplica y pasan a ser $(2N, 1)$, $(2K, 1)$, $(2N, 2K)$ y $(2N, 1)$, respectivamente.

Para la modulación de los datos transmitidos por los usuarios se hará uso de modulaciones QAM de *M* símbolos complejos con $\sqrt{\boldsymbol{M}}$ bits por símbolo.

Para el cálculo del canal AWGN utilizado este se ha definido mediante la relación señal ruido (*Signal-to-Noise Ratio, SNR*) siendo E_b la energía por bit, N_0 la densidad espectral de ruido, N_t la cantidad de antenas transmisoras y k la cantidad de bits por símbolo de la constelación utilizada.

$$\boldsymbol{SNR} = \frac{\boldsymbol{E_b}}{\boldsymbol{N_0}} \boldsymbol{N_t k}.$$

La variación del vector de ruido AWGN queda definida por N_0 y E_s

$$\boldsymbol{\sigma^2} = \frac{\mathbf{1}}{\boldsymbol{E_s}/\boldsymbol{N_0}}.$$

Las técnicas de detección pueden considerarse de dos formas. Por un lado, están las técnicas de detección que minimizan el coste computacional asumiendo un aumento en el BER, estos son denominados sub-óptimos, y como ejemplos se encuentran ZF (*Zero-Forcing*) o MF (*Matched Filter*). Por otro lado, como solución óptima al problema de detección se tendría la detección ML (*Maximum-Likelihood*), donde se realiza una búsqueda exhaustiva de todas las señales para encontrar la mejor solución, el algoritmo MLSD (*Maximum-Likelihood Sequence Detection*) es un ejemplo de un algoritmo óptimo donde se busca

$$\hat{\boldsymbol{x}}_{\boldsymbol{ML}} = \mathbf{arg}\,\boldsymbol{min}\ \|\boldsymbol{y} - \boldsymbol{Hx}\|_2^2.$$

4.2 Algoritmos de detección seleccionados

4.2.1 Optimized Coordinate Descent-based BOX Equalization

El algoritmo OCD-BOX [12] ha sido elegido ya que presenta una buena eficiencia junto con una baja complejidad. Además, según se explica en [11], la desventaja de este algoritmo es que se obtiene un bajo rendimiento cuando la relación entre antenas transmisoras y receptoras es cercana a 1, en este caso no supone ninguna desventaja debido a que se espera que la cantidad de drones sea muy inferior al número de antenas receptoras.

OCD-BOX es una relajación convexa de la condición ML a un cuadrado alrededor de la constelación empleada, reduciendo su coste computacional. En la Figura 18 se puede ver como se limita la zona de búsqueda. El problema de ecualización relajado del detector BOX es el siguiente:

$$\tilde{\boldsymbol{S}}_{\boldsymbol{w}}^{\ \boldsymbol{BOX}} = \mathbf{arg}\min_{\boldsymbol{z}\in\mathcal{C}_0^U} \|\boldsymbol{y}_{\boldsymbol{w}} - \boldsymbol{H}_{\boldsymbol{w}}\boldsymbol{z}\|_2^2. \quad (3)$$

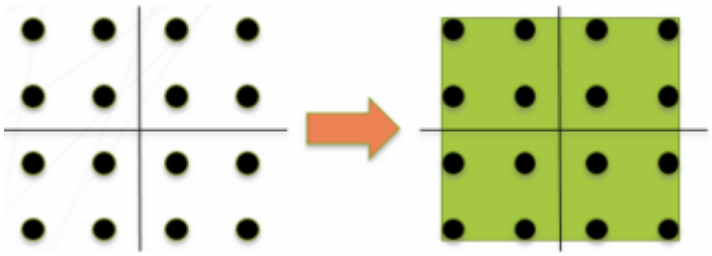


Figura 18: Relajación de la zona de búsqueda realizada en detección BOX.

OCD-BOX emplea el método de CD (*Coordinate Descent*), un método iterativo para encontrar el valor mínimo de una función dada. En cada iteración se evalúa la regla de selección y se selecciona el valor que la minimice. En este caso la regla de selección para OCD-BOX es la siguiente:

$$\hat{z}_u = \boldsymbol{proj}_{C_o}\left(\frac{1}{\|h_u\|_2^2}\boldsymbol{h}_u^H\left(\boldsymbol{y} - \sum_{j\neq u}\boldsymbol{h}_j\boldsymbol{z}_j\right)\right). \quad (4)$$

Mediante esta selección se obtiene el valor de las coordenadas en el nuevo plano, restringido a la zona de la constelación. La ecuación $proj_{Co}(\cdot)$ devolverá las coordenadas de entrada si estas se encuentran dentro del nuevo plano definido o, en el caso de que estas se encuentran fuera, las coordenadas más cercanas a las coordenadas del plano original que se encuentren dentro del nuevo plano

$$\boldsymbol{proj}_{C_o}(\boldsymbol{w}) = \begin{cases} \boldsymbol{w} & \boldsymbol{if\ w\ \in\ C_o} \\ \boldsymbol{\arg min}_{q\in C_o}|\boldsymbol{w}-\boldsymbol{q}| & \boldsymbol{if\ w\ \notin\ C_o} \end{cases}. \quad (5)$$

Al ser CD un proceso iterativo esto provoca que una parte de las operaciones se repitan durante el número de iteraciones establecidas. OCD-BOX realiza un preprocesado para reducir la cantidad de operaciones que se han repetir en cada iteración, haciendo que sea más eficiente. Se precomputan 2 variables, la inversa de las normas al cuadrado de las columnas de la matriz del canal (d_u^{-1}) y sus ganancias (p_u). En la Figura 19 se puede ver el pseudocódigo del algoritmo OCD-BOX.

Algorithm 1 Optimized Coordinate Descent (OCD)

inputs: $\mathbf{y}$, $\mathbf{H}$, and N_0
initialization:
  $\mathbf{r} = \mathbf{y}$ and $\mathbf{z}^{(0)} = \mathbf{0}^{U\times 1}$
  MMSE mode: $\alpha = N_0$ and $\mathcal{C} = \mathbb{C}$
  BOX mode: $\alpha = 0$ and $\mathcal{C} = \mathcal{C}_\mathcal{O}$
preprocessing:
  $d_u^{-1} = (\|\mathbf{h}_u\|_2^2 + \alpha)^{-1},\ u = 1,\ldots,U$
  $p_u = d_u^{-1}\|\mathbf{h}_u\|_2^2,\ u = 1,\ldots,U$
equalization:
for $k = 1,\ldots,K$ **do**
  for $u = 1,\ldots,U$ **do**
    $z_u^{(k)} = \text{proj}_\mathcal{C}\left(d_u^{-1}\mathbf{h}_u^H\mathbf{r} + p_u z_u^{(k-1)}\right)$
    $\Delta z_u^{(k)} = z_u^{(k)} - z_u^{(k-1)}$
    $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{h}_u \Delta z_u^{(k)}$
  end for
end for
outputs: $\tilde{\mathbf{s}} = [z_1^{(K)},\ldots,z_U^{(K)}]^T$

Figura 19: Pseudocódigo del detector OCD-BOX.

Para aplicar la detección OCD-BOX a los símbolos recibidos se ha implementado la función *BoxDetection_OCD()*. Esta recibe como entrada la matriz de datos recibida, la matriz del canal MIMO, los símbolos de la constelación M-QAM utilizada y el número de iteraciones (K). Los pasos a seguir son los que se han indicado en el pseudocódigo.

En primer lugar, se inicializan las variables a utilizar como el vector de la solución inicial, la cantidad de usuarios que transmiten o el valor máximo de los símbolos de la constelación utilizada. Seguidamente, se realiza el preprocesado de la inversa de la norma al cuadrado de las columnas de la matriz de canal (d) y sus ganancias (p).

Realizados estos pasos, se inicia el bucle que actualiza en cada iteración (El número de iteraciones viene dado como un parámetro de entrada) el valor de las coordenadas. Para realizar la actualización se ha definido la función *projeccion()* que realiza la comparación definida en (5). A continuación, en la Figura 20, se puede observar el pseudocódigo de la función.

Input: w,c
for $k=1,...,K$ **do**
  for $u=1,...,U$ **do**
    if $w_{u,k} \in c$ **then**
      $res_{u,k} \leftarrow w_{u,k}$
    else
      $res_{u,k} \leftarrow arg\ min_{q \in C_O} |w - q|$
    end
  end
end
Output: res

Figura 20: Pseudocódigo de la función Projeccion.

Como se puede observar en la Figura 21, la función solo actúa sobre los puntos que se encuentren fuera del nuevo plano convexo, los que se encuentren dentro de este no serán ajustados. La función tiene en cuenta los valores reales e imaginarios del símbolo, que son comparados con el máximo valor de la constelación (calculado previamente) para determinar si se ha de ajustar o no. La salida de la función son los símbolos actualizados.

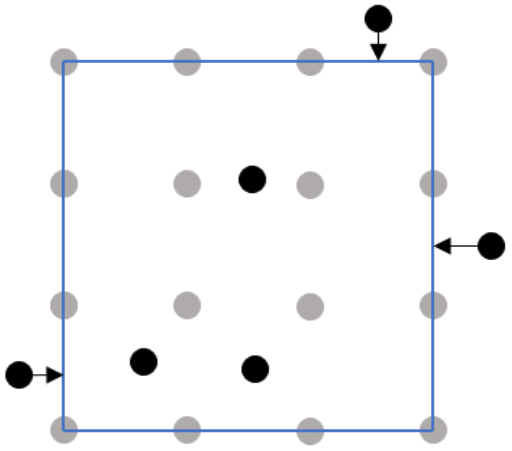


Figura 21: Ajuste de las coordenadas de fuera del plano.

Una vez obtenidas las nuevas coordenadas se calcula la diferencia con las anteriores para obtener la solución del problema de ecualización (3), la salida se toma como solución inicial para la próxima iteración. Por último, con los valores de las nuevas coordenadas se aplica una detección *hard* para ajustar los valores a posiciones de la constelación empleada.

4.2.2 ADMM-based Infinity Norm

Este segundo detector denotado como ADMIN [13] también realiza una detección basada en *BOX* igual que en OCD-BOX, por lo que presenta las mismas ventajas y desventajas que se han nombrado. A diferencia del anterior algoritmo que emplea CD, ADMIN emplea el método *ADMM* (*Alternating Direction Method of Multipliers*) para resolver los problemas de optimización convexos y no convexos. ADMM funciona rompiendo el problema original en subproblemas más pequeños que pueden resolverse de manera más eficiente. Para resolver el problema se utiliza la función de Lagrange, que es actualizada con la variable $\boldsymbol{\gamma}$. Para asegurar la convergencia de la función, $\boldsymbol{\gamma}$ ha de tener un valor comprendido entre $\mathbf{0} < \boldsymbol{\gamma} < \mathbf{1}$.

Para implementar este detector se ha definido la función *BoxDetection_ADMIN()*, como parámetros de entrada, la función recibe la matriz de datos recibida, la matriz del canal MIMO, la densidad espectral de potencia del ruido (**No**), la energía por símbolo (**Es**), el número de iteraciones del bucle de ecualización, los símbolos de la constelación M-QAM utilizada y el parámetro de actualización de la función de Lagrange ($\boldsymbol{\gamma}$).

El algoritmo de detección se divide en 3 partes. Primero se realiza un preprocesado que, en este caso, es el mismo que en OCD-BOX pero con más añadidos. Se calcula el parámetro de regularización $\boldsymbol{\beta}$ como $\boldsymbol{No/Es}$, y este es utilizado para calcular la matriz de filtrado MMSE (*Minimum Mean Square Error*) $\boldsymbol{W = H^H H + \beta I}$. En [13] se realiza la descomposición ldl de **G** (matriz de filtrado MMSE) para obtener una matriz correspondiente al triángulo inferior (**L**) y otra correspondiente a la diagonal (**D**).

Durante la fase de búsqueda de información, se encontraron implementaciones de algunos de los algoritmos utilizados, siendo ADMIN uno de ellos, en los que se realizaban variaciones respecto al algoritmo descrito en [13]. En [18] se realiza una implementación diferente de la ecualización para el algoritmo ADMIN, en lugar de calcular la descomposición ldl utiliza la factorización de Cholesky para obtener una matriz triangular inferior $\boldsymbol{L}$ de forma que $\boldsymbol{G = L' * L}$.

Con el objetivo de conocer si el BER que se obtiene para SNR dado es diferente dependiendo de que implementación se utilice, se han realizado pruebas para comparar ambas implementaciones. En el caso de la implementación de [18], esta obtiene un BER más bajo en comparación con la propuesta de [13] para un mismo valor de SNR. Es por esto que, en la implementación de este algoritmo, se ha realizado la factorización de Cholesky.

Seguidamente, se define la inicialización de las matrices **z** y $\boldsymbol{\lambda}$. Ambas se inicializan a cero con tantas filas como usuarios y con un número de columnas igual a la matriz de datos recibida.

Realizados estos apartados, se procede a realizar la ecualización tomando como solución inicial la ecualización MF. Al utilizar la factorización de Cholesky el cálculo de la matriz de datos **X** queda de la siguiente forma $\boldsymbol{X = L'\backslash(L\backslash(Y_{MF} + \beta(Z - \lambda)))}$. En segundo lugar, se aplica la función *projeccion()* para ajustar los valores que se encuentren fuera del plano definido por la constelación M-QAM. A la función se le pasa como parámetros de entrada la matriz **X**+$\boldsymbol{\lambda}$ y los M valores de la constelación. A continuación, la salida de la función se utiliza para actualizar el valor de $\boldsymbol{\lambda}$ para la siguiente iteración del bucle, $\boldsymbol{\lambda = \lambda - \gamma(\widehat{Z} - X)}$. Por último, se realizará una detección *hard* para ajustar los valores obtenidos a símbolos de la constelación utilizada.

Una vez terminado el bucle se realiza una detección *hard* para ajustar las coordenadas a puntos de la constelación utilizada. En la Figura 22 se muestra el pseudocódigo del algoritmo ADMIN.

inputs: $\mathbf{y}$, $\mathbf{H}$, N_0 and E_s
preprocessing
  $\beta = N_0 E_s^{-1} \epsilon$
  $\mathbf{G} = \mathbf{H}^H \mathbf{H} + \beta \mathbf{I}_U$
  $\mathbf{G} = \mathbf{L}\mathbf{d}\mathbf{L}^H$
  $\tilde{\mathbf{L}} = \mathbf{L}^{-1}$, $\tilde{\mathbf{D}} = \mathbf{D}^{-1}$
initialization
  $\mathbf{z} = \mathbf{0}$
  $\boldsymbol{\lambda} = \mathbf{0}$
detection
  $\mathbf{y}_{\text{MF}} = \mathbf{H}^H \mathbf{y}$
  for $i = 1 : K$
    $\hat{\mathbf{x}} \leftarrow \tilde{\mathbf{L}}^H \tilde{\mathbf{D}} \tilde{\mathbf{L}} (\mathbf{y}_{\text{MF}} + \beta(\mathbf{z} - \boldsymbol{\lambda}))$
    $\hat{\mathbf{z}} \leftarrow \text{proj}_{\mathcal{C}_\mathcal{O}}(\hat{\mathbf{x}} + \boldsymbol{\lambda}, \alpha)$
    $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \gamma(\hat{\mathbf{z}} - \hat{\mathbf{x}})$
    $\mathbf{z} \leftarrow \hat{\mathbf{z}}$
  end
output: $\hat{\mathbf{x}}$

Figura 22: Pseudocódigo del algoritmo ADMIN.

4.2.3 Successive Over Relaxation Method

Este algoritmo presenta la ventaja de que obtiene un rendimiento óptimo incluso con un ratio de antenas cercano a 1. Como desventaja, este algoritmo requiere que la matriz de Gram sea calculada de antemano y se introduzca como parámetro de entrada, lo que incrementa la complejidad. Además, se ha de seleccionar con cuidado el parámetro de relajación $\boldsymbol{\omega}$ ($\mathbf{0} < \boldsymbol{\omega} < \mathbf{2}$) para obtener un rendimiento óptimo. Al ser las carreras de drones un evento planificado (donde se realizan preparaciones previas a la carrera) estas desventajas no presentan un impacto tan grande.

SOR [14] es un método iterativo empleado para solucionar sistemas lineales unidimensionales que también puede ser utilizado para la resolución de sistemas lineales multidimensionales. En el caso de ser aplicado a un sistema multidimensional, el resultado se encuentra en una región de convergencia definida por el parámetro de relajación $\boldsymbol{\omega}$. El valor de $\boldsymbol{\omega}$ que se busca es aquel con el que se obtiene una convergencia óptima. Una condición necesaria para utilizar el método SOR es que la matriz A del sistema sea positiva, definida y simétrica.

$$\boldsymbol{x}^{(i+1)} = \left(\boldsymbol{L_A} + \frac{\mathbf{1}}{\boldsymbol{\omega}}\boldsymbol{D_A}\right)^{-\mathbf{1}} \left[\left(\left(\frac{\mathbf{1}}{\boldsymbol{\omega}} - \mathbf{1}\right)\boldsymbol{D_A} - \boldsymbol{L_A^T}\right)\boldsymbol{x}^{(i)} + \boldsymbol{b}\right]. \quad (6)$$

El algoritmo de detección basado en SOR implementado se basa en el hecho de que, a diferencia de MIMO convencional, las columnas de la matriz de canal de mMIMO son asintóticamente ortogonales. De esta manera se determina que la matriz Gram ($\mathbf{G}$) es positiva y definida. Además, el modelo de sistema empleado en el detector se ha convertido a números reales, lo que hace que la transpuesta y transpuesta conjugada de $\mathbf{G}$ sean simétricas ($\boldsymbol{G = H^H H = H^T H = G}$). Al cumplirse estas tres propiedades se puede utilizar el método SOR en el detector. Las iteraciones del detector se han descrito tal como se puede observar en la ecuación 4, definidas originalmente en [15].

La implementación del detector SOR se ha realizado mediante la función *sorMethod()*. Esta recibe como entrada la matriz de datos recibidos, la matriz de canal, el parámetro de regularización $\boldsymbol{\beta}$, el número de iteraciones del bucle de ecualización, el parámetro de relajación $\boldsymbol{w}$ y los valores de la constelación utilizada.

Como se ha realizado en [13], el algoritmo opera en números reales. En primer lugar, se realiza un preprocesado donde se las matrices de entrada son convertidas a matrices de números reales y se calcula la matriz de filtrado MMSE **W**. En segundo lugar, se realiza la inicialización de variables, donde se calcula la solución inicial con ecualización MF y se descompone la matriz de filtrado MMSE en la matriz diagonal, estrictamente superior e inferior $\boldsymbol{W = D + L + U}$. Por conveniencia, se define $\boldsymbol{w = \frac{1}{w}}$.

A continuación, se realiza la ecualización ejecutando el bucle tantas iteraciones como se haya indicado, dentro del bucle solo se ejecuta (6). Una vez terminada la ecualización se realiza una detección *hard* para ajustar los valores obtenidos a símbolos de la constelación utilizada. Los resultados que se devuelven son previamente convertidos de números reales a complejos. En la Figura 23 se presenta el pseudocódigo de la función implementada en Matlab.

Input: y,H,β, K, w

Preprocesado

$\mathrm{y} = \begin{bmatrix} \Re\{y\} & \Im\{y\} \end{bmatrix}^T$

$\mathrm{H} = \begin{bmatrix} \Re\{H\} & -\Im\{H\} \\ \Im\{H\} & \Re\{H\} \end{bmatrix}_T$

$\mathrm{W} := \mathrm{H}^T * H + \beta I$

Inicialización

$\mathrm{y}_{MF} := H^T * y$

w = 1/w

D := diag{W}

L := tril{W}

Up := triu{W}

Equalización

for $1,\ldots,K$ **do**

| x = (w*D+L)$^{-1}$(y_{MF} + ((w − 1) * D − Up) * x)

end

Output: x

Figura 23: Pseudocódigo del algoritmo SOR en Matlab.

4.2.4 Sequential Likelihood Ascent Search

Los detectores basados en búsqueda local, o LS (*Local Search*), reducen el coste ML en un vecindario dado que, dependiendo del tamaño, la complejidad aumentará o disminuirá.

SLAS es un algoritmo propuesto a partir de LAS (*Likelihood Ascent Search*). LAS se basa en, a partir de una solución inicial, buscar en el vecindario a su alrededor una solución que mejore el coste ML. Se han propuesto diferentes variaciones del algoritmo LAS, unas centrándose en reducir la complejidad mediante la reducción del tamaño del vecindario en el que se buscan las soluciones, y otras se han centrado en mejorar el BER evitando restringir la búsqueda a un vecindario fijo. SLAS [16] trata de juntar las ventajas de estas variaciones aunando la búsqueda a vecindarios reducidos y permitiendo que la búsqueda se amplíe a varios vecindarios.

Un vecindario está constituido por todas las combinaciones que difieren en **L** símbolos de un vector de tamaño $\mathbf{2N_t}$. Por lo tanto, se tendrá un total de $\binom{\boldsymbol{2N_t}}{\boldsymbol{L}}$ combinaciones para un **L** dado, este conjunto es denominado $\boldsymbol{I_k}$. Esta cantidad se ha de multiplicar por la cantidad de valores que puede tomar cada elemento del vector, que vendrá dado por el tamaño de constelación utilizado $\left(\sqrt{\boldsymbol{M}} - \mathbf{1}\right)^{\boldsymbol{L}}$. Esto da un total de $\left(\sqrt{\boldsymbol{M}} - \mathbf{1}\right)^{\boldsymbol{L}} \binom{\boldsymbol{2N_t}}{\boldsymbol{L}}$ combinaciones posibles para un vecindario de tamaño **L**.

La implementación de este detector se ha realizado mediante la función *SLAS()*. Esta función recibe como parámetros de entrada la matriz de datos recibida, la matriz de canal, los valores de

la constelación utilizada, el parámetro de regularización $\boldsymbol{\beta}$, el tamaño máximo de vecindario hasta el cual se realiza la búsqueda y la regla de selección utilizada.

El algoritmo SLAS se divide en 4 partes. En primer lugar, se han de determinar los sets de índices a actualizar. Como métrica a aplicar en las reglas de selección se utiliza $\boldsymbol{u_k = \sum_{s=1}^{L} f_{i_s}^2 \ \forall \ I_k}$, donde se realiza el sumatorio de $\boldsymbol{f_i = \frac{(y-Hx^r)^T h_i}{\|h_i\|}}$ ($\boldsymbol{x^r}$ corresponde al vector de la iteración actual), que corresponden con los L valores actualizables de cada combinación. Esto significa que para un vecindario de tamaño L=2, en cada combinación habrá 2 valores de $\boldsymbol{f_i}$ a tener en cuenta para el sumatorio de $\boldsymbol{u_k}$. Como se puede observar en la Figura 24 para un caso con 3 antenas transmisoras y **L**=2 se tienen $\boldsymbol{\binom{2N_t}{L} = 15}$ combinaciones posibles de valores actualizables.

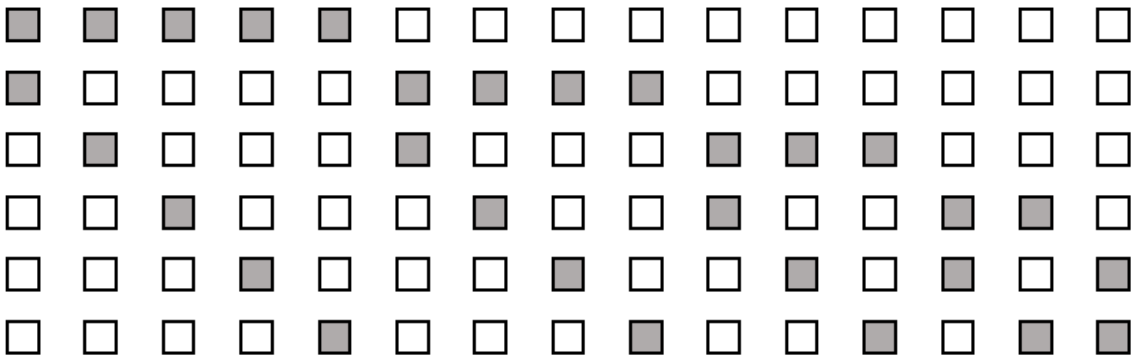


Figura 24: Representación de los valores actualizables en cada combinación para un tamaño L=2.

Una vez calculados todos los $\boldsymbol{u_k}$ para cada combinación del set $\boldsymbol{I_k}$ se aplica una de las reglas de detección para obtener un vecindario reducido, que será el utilizado para realizar las actualizaciones del coste ML. En [15] se proponen 3 estrategias diferentes para seleccionar los $\boldsymbol{u_k}$.

K sets más probables: Se selecciona un conjunto de K sets con los valores de $\boldsymbol{u_k}$ más elevados, limitando la búsqueda solo a estos sets. La selección del valor de K es muy importante ya que un número demasiado elevado conllevará un aumento de la complejidad y un valor demasiado bajo puede degradar el rendimiento del detector. Un valor apropiado para K es en torno al 10% de $\boldsymbol{\binom{2N_t}{L}}$.

Selección basada en normalización: Se selecciona un conjunto de K sets más probables realizando una normalización entre el máximo valor de los diferentes $\boldsymbol{u_k}$. Los resultados se encuentran entre 0 y 1, se elegirán los $\boldsymbol{u_k}$ con valores iguales o superiores a 0.8

$$\tilde{u}_k = \frac{u_k}{u_{max}} \ \forall k = 1, 2, \dots, \binom{2N_t}{L}. \tag{7}$$

Selección basada en normalización de suma de cuadrados: Similar a la anterior regla de selección, aquí también se realiza una normalización, pero en este caso se busca que la suma de los $\boldsymbol{u_k}$ seleccionados sea mayor que un cierto valor umbral

$$\tilde{u}_k = \frac{u_k}{\sqrt{\sum_k |u_k|^2}} \ \forall k = 1, 2, \dots, \binom{2N_t}{L}. \tag{8}$$

Para el cálculo del BER se ha hecho uso de la selección basada en normalización.

Una vez se ha aplicado la regla de selección elegida se actualizan los sets de índices seleccionados. Para cada uno de ellos se calcula $\boldsymbol{\eta^* = \left\lceil \frac{\left(H_{I_k}^T H_{I_k}\right)^{-1} H_{I_k}^T e^r}{d_{min}} \right\rceil}$, donde $\boldsymbol{e^r = y - Hx^r}$ corresponde al vector

de error de la iteración actual. η^* tendrá un tamaño de $(2N_T, 1)$, igual que el vector a actualizar, y según el set $\boldsymbol{I_k}$ que se esté actualizando se actualizarán unas posiciones u otras. El valor actualizado será $\tilde{\boldsymbol{x}}_{\boldsymbol{i}} = [\boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{r}} + \boldsymbol{\eta}_{\boldsymbol{i}}^{*}\boldsymbol{d}_{\boldsymbol{min}}]$. En la Figura 25 se puede ver como sería la actualización de los valores del vector para un set de I_k dado cuando el tamaño de vecindario es **L**=2.

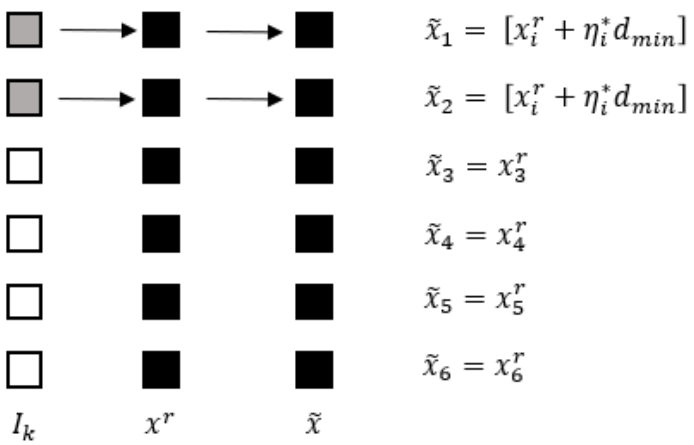


Figura 25: Ejemplo de actualización de un set de índices dado.

Una vez realizada la actualización se comprueba su coste ML, en el caso de que fuera inferior al coste actual se guardan los valores actualizados, ya que es la solución que da mejor resultado. Esta operación de actualización que se ve en la Figura 25 se realiza para cada uno de los sets de $\boldsymbol{I_k}$, de esta forma se obtiene la solución que requiere un menor coste ML de todos los sets del vecindario reducido. Por último, se actualiza el tamaño de vecindario para la siguiente búsqueda. Si el coste calculado es inferior al de la iteración previa el tamaño del vecindario se reduce a **L**=1, si fuera mayor, se incrementa en uno el tamaño del vecindario. El bucle termina cuando el tamaño de vecindario supera un valor máximo. En la Figura 26 se expone el pseudocódigo del algoritmo SLAS.

Input : $\mathbf{y}$, $\mathbf{H}$, Ω, max_{stage}
Output: $\hat{\mathbf{x}}$

Initialization $r = 0$ & $L = 1$;
$\mathbf{x}^r \leftarrow$ output of MF/ZF/MMSE detector;
$Cost_{next} \leftarrow \|\mathbf{y} - \mathbf{H}\mathbf{x}^r\|^2$ & $Cost_{pre} \leftarrow \infty$;
while $L \leq max_{stage}$ **do**
  Determining the sets of indices to be updated
  $f_i = \frac{(\mathbf{y}-\mathbf{H}\mathbf{x}^r)^T \mathbf{h}_i}{\|\mathbf{h}_i\|}$ $\forall i = 1, 2, \ldots 2N_t$;
  Generate all possible combinations of L indices and save it as I_k where $k = 1, 2, \ldots, \binom{2N_t}{L}$;
  $u_k = \sum_{i \in I_k} f_i^2$ $\forall k = 1, 2, \ldots, \binom{2N_t}{L}$;
  Apply the chosen selection rule on u_k's and select few I_k's;
  $Cost_{pre} \leftarrow Cost_{next}$;
  $\mathbf{x}_{temp} \leftarrow \mathbf{x}^r$;
  Updating the selected sets of indices
  for $j \leftarrow 1$ **to** *number of selected* I_k*'s* **do**
    Select one set of indices I_k and proceed;
    $\eta_{I_k}^* = \left\lceil \frac{\left(\mathbf{H}_{I_k}^T \mathbf{H}_{I_k}\right)^{-1} \mathbf{H}_{I_k}^T \mathbf{e}^r}{d_{min}} \right\rfloor$;
    for $i \leftarrow 1$ **to** $2N_t$ **do**
      if $i \in I_k$ **then**
        $\tilde{x}_i = [x_i^r + \eta_i^* d_{min}]$;
      else
        $\tilde{x}_i = x_i^r$;
      end
    end
    Updating the best available vector
    $Cost_{temp} = \|\mathbf{y} - \mathbf{H}\tilde{\mathbf{x}}\|^2$;
    if $(Cost_{temp} < Cost_{next})$ **then**
      $\mathbf{x}_{temp} \leftarrow \tilde{\mathbf{x}}$;
      $Cost_{next} = Cost_{temp}$;
    end
  end
  Determining the neighborhood search size
  if $(Cost_{next} < Cost_{pre})$ **then**
    $L = 1$;
  else
    $L = L + 1$;
  end
  $\mathbf{x}^{r+1} \leftarrow \mathbf{x}_{temp}$;
  $r = r + 1$;
end
return $\hat{\mathbf{x}} \leftarrow \mathbf{x}^{r+1}$.

Figura 26: Pseudocódigo del algoritmo SLAS.

Capítulo 5. Pruebas realizadas

5.1 Condiciones de las pruebas

Los algoritmos OCD-BOX, ADMIN, SOR y SLAS, seleccionados como los más adecuados para el contexto de una carrera de drones, se han implementado en Matlab con el objetivo de evaluar la tasa de error de bit, o BER, en función del ratio señal/ruido, o SNR (*Signal to Noise Rate*), y el coste computacional en función de los flops requeridos, para así conocer que algoritmo ofrecería el mejor rendimiento al ser implementado en una carrera de drones.

Las carreras normalmente se realizan en pabellones polideportivos o estadios de futbol. Los circuitos recorren el interior de las instalaciones, por lo que no se tendrá LOS durante gran parte de la carrera. Debido a esto, para la simulación de los 4 canales de cada algoritmo se ha optado por utilizar un canal de tipo Rayleigh, para así aprovechar las componentes multicamino. En la Figura 27 se puede observar un ejemplo en el que se ha tomado una representación en 3D del estadio de futbol Mestalla [17] y, empleando la técnica de trazado de rayos de Matlab, se ha podido comprobar que es posible aprovechar las componentes multicamino. Esta es la razón por la cual se ha utilizado un canal Rayleigh y no Rician.

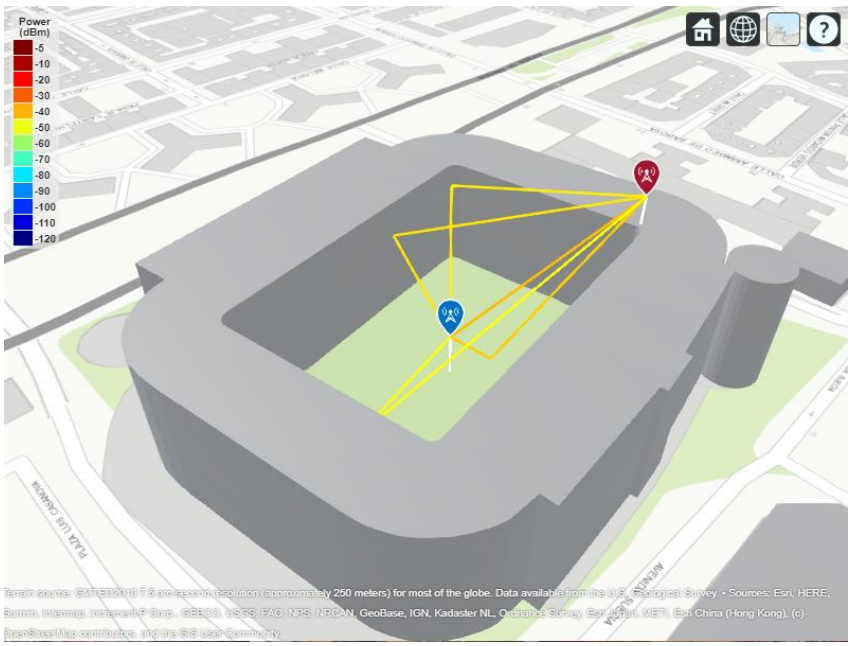


Figura 27: Representación de trazado de rayos en Matlab.

En el ejemplo de la Figura 27 se ha tenido en cuenta un espacio que no estaría completamente cerrado ya que no hay un techo que lo cubra. En el caso de un polideportivo cerrado las componentes multicamino aumentarán.

mMIMO es una de las nuevas tecnologías que entran con el estándar 5G. Para la simulación de la información se han utilizado las modulaciones 16-QAM y 64-QAM junto con OFDM. Aunque en 5G se plantea el uso de 256-QAM, no se ha tenido en cuenta para estas simulaciones ya que esta modulación añadiría una mayor complejidad y el rendimiento no aumentaría en proporción para los bitrates que se busca transmitir.

En el estándar 5G, los anchos de banda utilizados aumentarán de tamaño, llegando hasta los 100 Mhz. Para realizar las simulaciones se ha optado por utilizar un ancho de banda de 50 Mhz, el cual cuenta con un tamaño de FFT de 5120 subportadoras, de las cuales 3010 están ocupadas por señales de datos o pilotos y 2110 son nulas (guarda). En la tabla 2 se pueden ver los diferentes valores de NFFT, subportadoras ocupadas y nulas en función del ancho de banda que se utilice. En la simulación, del total de portadoras ocupadas, se han introducido señales piloto con una separación de 15 puntos entre cada una.

	5 Mhz	**10 Mhz**	**15 Mhz**	**20 Mhz**	**50 Mhz**	**100 Mhz**	**200 Mhz**
NFFT	512	1024	1536	2048	5120	1024	20480
Ocupadas	301	601	901	1201	3010	6010	12010
Nulas	211	423	635	847	2110	4230	8470

Tabla 2: Diferentes valores de subportadoras en función del ancho de banda.

La cantidad de datos transmitidos en cada simulación es de 50.000 bits. Dependiendo de la modulación QAM que se utilice, se generará la cantidad adecuada de símbolos QAM y OFDM para transmitir dicha cantidad de bits. Primero se determina el total de portadoras de datos que se tienen y se multiplica por el número de bits/símbolo de la modulación QAM empleada, obteniendo la cantidad de bits por símbolo OFDM. Seguidamente se divide el total de bits a transmitir entre esta cantidad calcula (redondeada a un valor entero) para obtener el total de símbolos OFDM que se van a transmitir. Con estos datos calculados se genera una matriz en 3 dimensiones de números aleatorios, siendo la primera dimensión el número de portadoras de datos, la segunda el número de símbolos OFDM a transmitir y la tercera la cantidad de antenas transmisoras. Los números aleatorios generados son enteros que se encuentran entre 0 y M-1. Esta matriz es introducida en la función *qammod()* de Matlab para generar la secuencia de símbolos M-QAM. En la Figura 28 se puede observar como se ha realizado este proceso en Matlab.

```
%% generar datos a transmitir

Nbits = NdataOFDM*k;
NsymOFDM = ceil(NbitsTx/Nbits);
dataTx = randi([0 M-1],NdataOFDM,NsymOFDM,aTx);
symTx = qammod(dataTx, M,'gray');
```

Figura 28: Datos M-QAM generados.

Además de los símbolos de datos también se generan y modulan los símbolos de las señales piloto. A diferencia de los datos transmitidos, la modulación empleada es BPSK y para ello se utiliza la función *pskmod()* para generar los símbolos. De esta manera se consigue que los símbolos piloto tengan una mayor energía que los de datos. Como se puede ver en la Figura 29, el procedimiento es similar a la generación de los símbolos de datos.

```
%% Generar señales piloto

pilotTx = randi([0 Modpilot-1], NpilotOFDM*NsymOFDM*aTx,1);
symPilot = pskmod(pilotTx, Modpilot, 0, 'gray');
symPilot = reshape(symPilot,NpilotOFDM,NsymOFDM,aTx);
```

Figura 29: Generación de símbolos piloto.

Los símbolos M-QAM y BPSK generados se modulan en OFDM para ser transmitidos. La modulación se realiza mediante la función *ofdmmod()* de Matlab, esta toma como entrada los símbolos de datos y pilotos, la distribución en frecuencia de los datos, nulos y pilotos y el tamaño de prefijo cíclico. La distribución de las nulas se ha hecho de forma que se han introducido la mitad en el lado izquierdo, una portadora central (la componente de DC) y a la derecha la mitad menos uno. Se ha hecho de esta forma ya que la cantidad de nulas es par. En la Figura 30 se muestra como quedaría la distribución de las subportadoras en frecuencia.

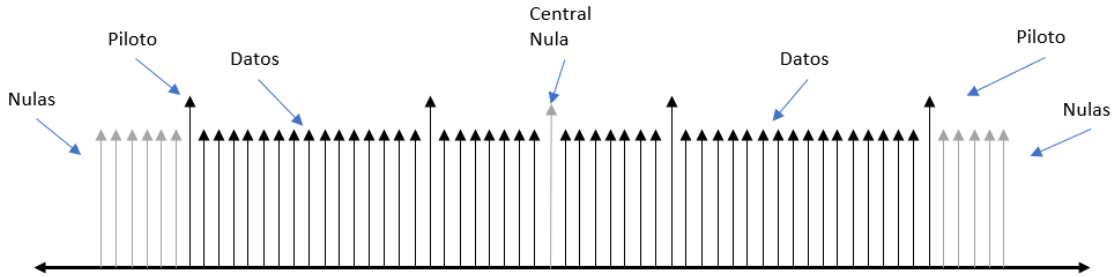


Figura 30: Representación de la distribución de la señal en frecuencia.

Como salida del modulador se obtiene una matriz con una cantidad de columnas igual al número de antenas transmisoras, esta matriz será transmitida por el canal MIMO. En la Figura 31 se muestra como se ha utilizado la función *ofdmmod()* para generar la señal.

```
%% Generar señal OFDM

pilotidx = [linspace((NGuard/2)+1,NFFT/2,NpilotOFDM/2)
    linspace((NFFT/2)+2,NFFT-(NGuard/2)-1,NpilotOFDM/2)]';
nullidx = [1:(NGuard/2)  (NFFT/2)+1 NdataPilotOFDM+(NGuard/2)+2:NFFT]';
Tx = ofdmmod(symTx,NFFT,Ncp,nullidx,round(pilotidx),symPilot).';
```

Figura 31: Generación de señal OFDM.

El canal está compuesto por un canal MIMO y un canal de ruido AWGN. Para introducir el canal MIMO se ha creado una matriz de números aleatorios complejos con las dimensiones de las antenas transmisoras y receptoras. En el caso del canal AWGN se crea un canal utilizando la función *comm.AWGNChannel()* del módulo de comunicaciones de Matlab. Los parámetros que se utilizan para su creación son el valor de SNR utilizado en la iteración actual de la simulación y la potencia media de la señal, que es calculada como el valor medio de la señal al cuadrado. A través del canal AWGN se pasa la señal multiplicada por la matriz de canal MIMO, la implementación se muestra en la Figura 32.

```
% Rayleigh distributed channel response matrix
H = complex(randn(aRx, aTx),randn(aRx, aTx))*sqrt(0.5);

% AWGN Channel
AveragePower = mean(abs(Tx).^2);
awgnchan = comm.AWGNChannel('NoiseMethod',...
    'Signal to noise ratio (SNR)', 'SNR', SNR(i),...
    'SignalPower', mean(AveragePower));
MIMORx = awgnchan(H*Tx);
```

Figura 32: Transmisión de la señal a través del canal MIMO y AWGN.

En recepción, se demodula la señal OFDM para extraer las portadoras de datos mediante la función *ofdmdemod()*. La matriz obtenida se encuentra en 3 dimensiones, por lo que se ha de transformar en una matriz de 2 dimensiones para poder introducirla en los detectores. Dependiendo del detector que se utilice, será necesario realizar el cálculo previo de parámetros como, por ejemplo, la energía por símbolo.

Una vez aplicado el detector a los símbolos M-QAM recibidos estos serán demodulados y se realizará el cálculo del BER. Para cada valor de SNR elegido se utiliza el método de Montecarlo para obtener el valor medio del BER. En la Figura 33 se puede observar un diagrama de todo el proceso.

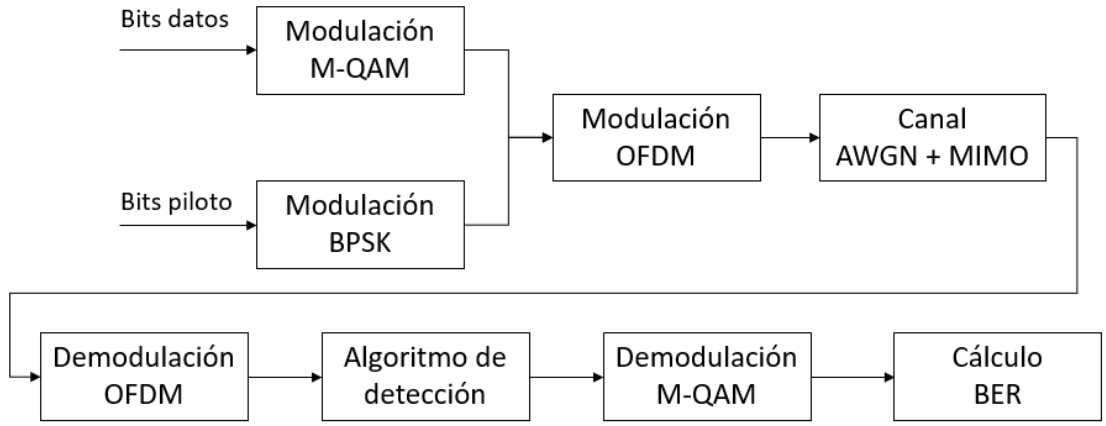


Figura 33: Diagrama de la simulación realizada.

Las pruebas se realizan con diferentes configuraciones de antenas para evaluar como influye en el BER el ratio entre antenas transmisoras y receptoras. Se han escogido un total de cuatro configuraciones: 50x10, 100x10, 100x20 y 128x16. Aunque 50x10 y 100x20 mantienen el mismo ratio, se han escogido ambas para observar que diferencias se encuentran cuando, en comparación con 100x10, se aumentan las antenas transmisoras o las receptoras. Para todas las configuraciones se han realizado las pruebas utilizando modulaciones 16-QAM y 64-QAM.

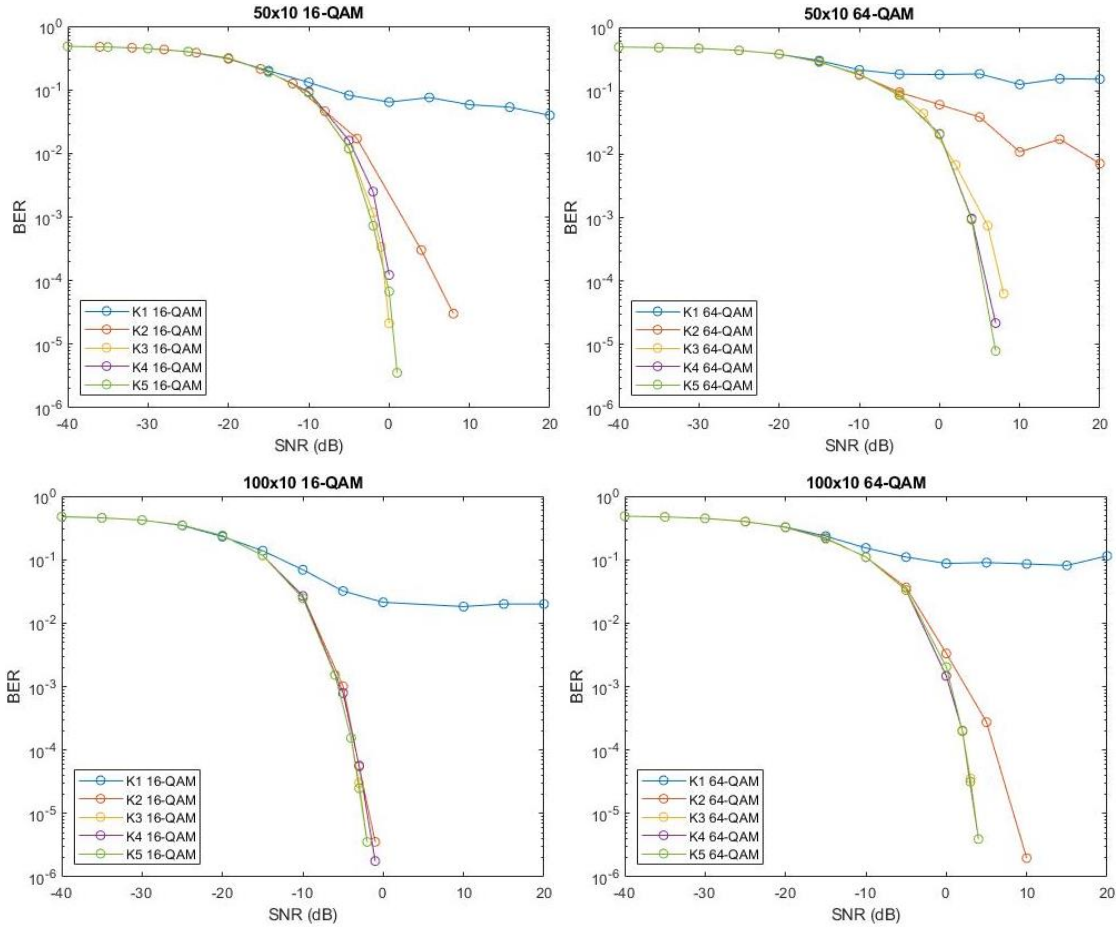
Los algoritmos OCD-BOX, ADMIN y SOR se han ejecutado con iteraciones de 1 a 5 (K1, K2, K3, K4 y K5), ya que en [13], [14] y [15] las iteraciones que se realizan entran dentro de este rango, permitiendo así realizar comparaciones entre los resultados obtenidos. En cuanto al algoritmo SLAS, se han realizado las simulaciones con un tamaño máximo de vecindario de búsqueda de 1 y 2 (L1 y L2). Debido a la búsqueda iterativa que realiza este algoritmo, solo se ha realizado con estos dos valores, ya que tamaños mayores tienen un gran coste computacional, provocando que las simulaciones requieran de un tiempo inasumible para realizarse.

Todas las simulaciones para el cálculo del BER se han realizado en un servidor perteneciente al Grupo de Tratamiento de Audio y Comunicaciones (GTAC) del Instituto de Telecomunicaciones y Aplicaciones Multimedia (iTEAM).

5.2 Estudio del BER de los algoritmos

5.2.1 Simulaciones OCD-BOX

Los resultados obtenidos de las simulaciones de OCD-BOX muestran que, para obtener un resultado de BER aceptable, se ha de realizar un mínimo de 2 iteraciones. En el caso de realizar una iteración, se obtiene que, para todas las configuraciones, el BER tiende a mantenerse casi constante en valores altos de SNR. Este resultado se puede observar en la Figura 34, donde para K1 no se obtienen valores inferiores a 10^{-2}. Por lo tanto, utilizar OCD-BOX con solo una iteración no ofrece un resultado que adecuado para implementarlo en un detector.



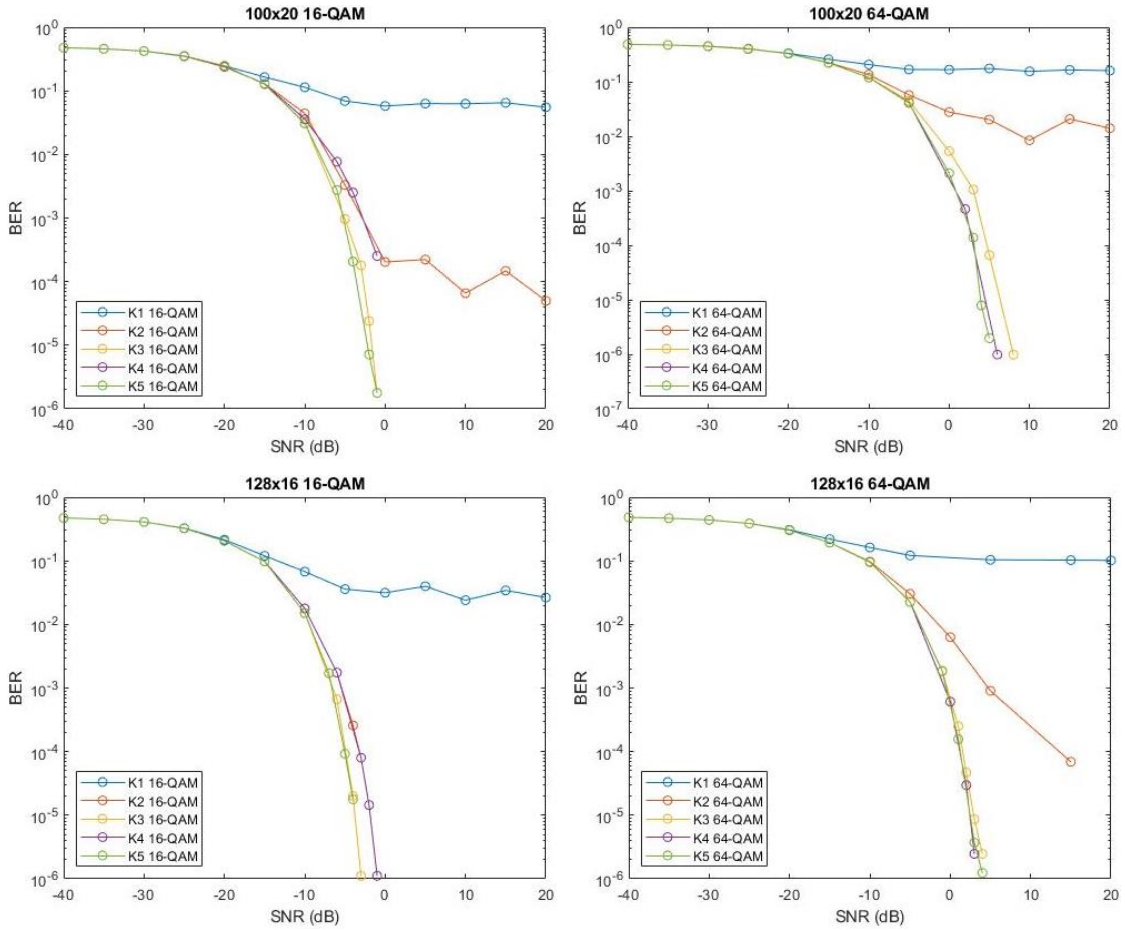


Figura 34: Rendimiento en términos de BER del algoritmo OCD-BOX para las diferentes configuraciones seleccionadas.

Al realizar dos iteraciones, las configuraciones 100x10 y 128x16 consiguen obtener un BER por debajo de 10^{-4} para ciertos valores de SNR no muy elevados. Sin embargo, en las configuraciones 100x20 y 50x10 los resultados siguen el mismo patrón que cuando se realiza una iteración, a excepción de 50x10 con modulación 16-QAM. Como se puede observar en la Figura 34, en 100x10 y 128x16 se puede obtener un BER inferior a 10^{-4} para 16-QAM a partir de -3 dB. En el caso de la modulación 64-QAM, se necesita un valor superior a 6 dB en 100x10 y superior a 13 dB en 128x16 para que el BER sea inferior a 10^{-4}. En comparación con estas dos modulaciones, la configuración 50x10 tiene un rendimiento inferior en 16-QAM, ya que para obtener un BER inferior a 10^{-4} se requiere un SNR de 6 dB, lo mismo que en 100x10, pero con una modulación de mayor orden.

En este caso es posible utilizar OCD-BOX con solo dos iteraciones cuando las antenas receptoras superan a las transmisoras por un factor de ocho o superior. Como se ha visto, cuando la relación entre antenas es de un factor de cinco, se podrá conseguir un BER óptimo cuando las antenas transmisoras sean diez, o menos, y se utilice una modulación 16-QAM, a costa de requerir un SNR más elevado.

Al realizar las simulaciones con tres iteraciones es cuando todas las configuraciones de antenas muestran un resultado en el que el BER llega a estar por debajo de 10^{-4}, tanto en 16-QAM como en 64-QAM. En la Figura 34 se puede observar que con K3 los resultados obtenidos muestran que se alcanza un BER inferior a 10^{-4}, en el peor de los casos, con un SNR que no supera los 10 dB.

Con la modulación 16-QAM, las configuraciones 50x10, 100x10, 100x20 y 128x16 alcanzan un BER de 10^{-4} con un SNR de -0.5 dB, -3.75 dB, -2.71 dB y -4.91 dB, respectivamente. La configuración 100x20 es la que ha presentado una mayor mejora respecto a la iteración anterior, ya que ahora el BER tiende a descender conforme se aumenta la SNR, en vez de mantenerse constante. La configuración 50x10 también presenta una gran mejora respecto a la iteración anterior, pasando de 6 dB a -0.5. Por último, las configuraciones 100x10 y 128x16 no han visto afectado su rendimiento de la misma forma que las otras configuraciones, pero este sí que ha mejorado. En el caso de 100x10 la reducción de la SNR ha sido muy pequeña, de -0.75 dB, y en 128x16 este ha disminuido -1.91 dB.

En el caso de la modulación 64-QAM, se puede observar que las cuatro configuraciones sí que consiguen que el BER disminuya con forme aumenta la SNR. En esta iteración se alcanza un BER de 10^{-4} con 7.62 dB en 50x10, 2.41 dB en 100x10, 4.7 dB en 100x20 y 1.55 dB en 128x16. 50x10 y 100x20 presentan una mayor reducción que el resto de las configuraciones, ya que el BER ya no se mantiene constante. Sin embargo, la configuración 128x16 presenta una gran reducción al pasar de 13 dB a 1.55 dB, 11.45 dB de diferencia.

En la ejecución de OCD-BOX con cuatro iteraciones se obtiene que, para 16-QAM, se alcanza un BER de 10^{-4} con 0 dB en 50x10, -3.43 dB en 100x10, 0 dB en 100x20 y -3.22 dB en 128x16. En cuanto a 50x10 y 100x20, ambas configuraciones obtienen un BER a partir de 0 dB antes de llegar a 10^{-4}, debido a esto, con cuatro iteraciones estas configuraciones serían las que ofrecen un mejor resultado. Sin embargo, todas las configuraciones han empeorado su rendimiento respecto a la iteración anterior, ya que con los mismos valores de SNR en K3 se obtiene un mejor BER.

En cuanto a 64-QAM, se obtiene un BER de 10^{-4} con 5.8 dB en 50x10, 2.35 dB en 100x10, 3 dB en 100x20 y 1.2 dB en 128x16. Las cuatro configuraciones mejoran el rendimiento en comparación con la iteración anterior. Sin embargo, esta mejora es muy pequeña.

Por último, se han realizado simulaciones de las cuatro configuraciones con un total de cinco iteraciones en el detector. Para 16-QAM se obtiene un BER de 10^{-4} con -0.33 dB en 50x10, -3.76 dB en 100x10, -3.57 dB en 100x20 y -5.055 dB en 128x16. En esta iteración todas las configuraciones han mejorado, pero muy ligeramente.

En 64-QAM se alcanza un BER 10^{-4} con 5.41 dB en 50x10, 2.37 dB en 100x10, 3.11 dB en 100x20 y 1.25 dB en 128x16. La variación en rendimiento que se produce es mínima en comparación con la iteración anterior y, excepto en la configuración 50x10, el SNR para obtener un BER 10^{-4} aumenta.

De los resultados de las cinco iteraciones diferentes cabe destacar que: para que el detector OCD-BOX funcione se han de realizar un mínimo de dos iteraciones. Al aumentar las iteraciones por encima de dos, el rendimiento mejora tanto para 16-QAM como para 64-QAM, siendo esta última la mayor beneficiada al aumentar las iteraciones. Esto se puede observar en 128x16, donde en 64-QAM se pasa de 13 dB en 2 iteraciones a 1.25 dB en 5 iteraciones. En la Tabla 3 se puede observar la evolución del BER entre las diferentes iteraciones.

OCD-BOX		Número de iteraciones 1	2	3	4	5
50x10	16-QAM	--	6	-0.5	0	-0.33
	64-QAM	--	--	7.62	5.8	5.41
100x10	16-QAM	--	-3	-3.75	-3.43	3.76
	64-QAM	--	6	2.41	2.35	2.37
100x20	16-QAM	--	--	-2.71	0	-3.57
	64-QAM	--	--	4.7	3	3.11
128x16	16-QAM	--	-3	-4.91	-3.22	-5.05
	64-QAM	--	13	1.55	1.2	1.25

Tabla 3: Diferentes valores de SNR (dB) para los cuales se obtiene un BER igual a 10^{-4} en función de la configuración, modulación e iteración.

5.2.2 Simulaciones ADMIN

En este algoritmo también se han realizado simulaciones de una a cinco iteraciones. En ADMIN, el valor del parámetro γ está comprendido en el rango $\mathbf{0 < \gamma < 1}$. Para conocer que efecto tiene el valor de γ en el cálculo del BER, se han realizado tres simulaciones variando el $\boldsymbol{\gamma}$ entre 0.25, 0.5 y 0.75. En la Figura 35 se puede observar la variación en el BER obtenido con diferentes parámetros de $\boldsymbol{\gamma}$. Debido a que esta variación es mínima, se ha optado por utilizar $\boldsymbol{\gamma = 0.5}$ para todas las simulaciones.

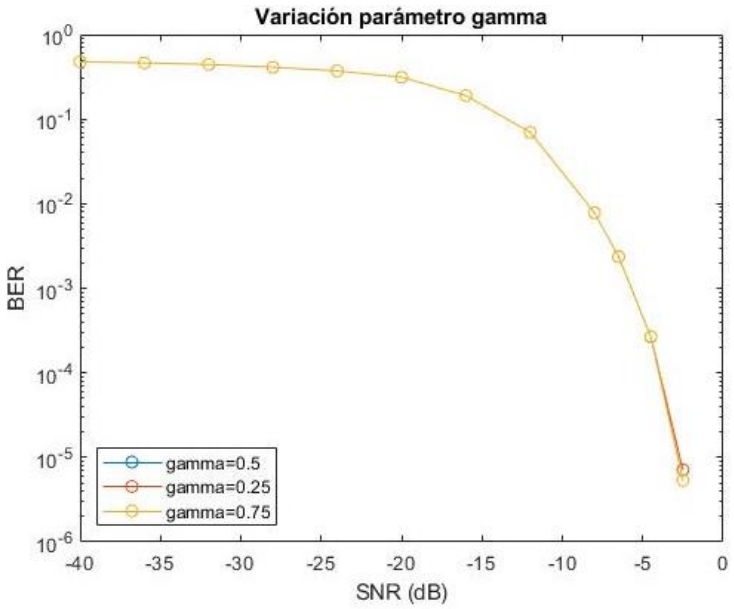
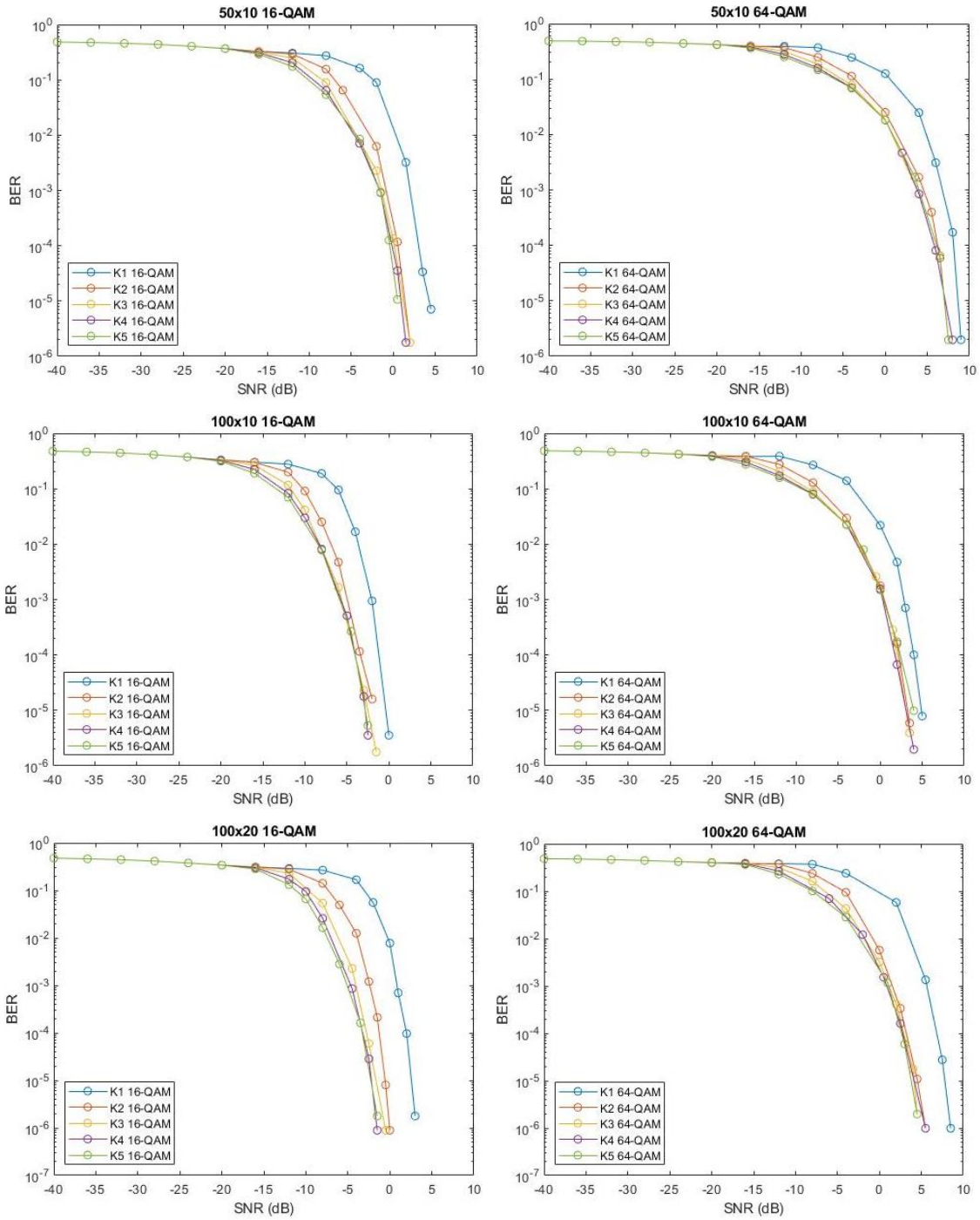


Figura 35: Comparación de BER al variar el parámetro γ.

Al ejecutar el algoritmo con una iteración se puede observar que, en 16-QAM, para alcanzar un BER de 10^{-4} se requiere una SNR de 3.02 dB en 50x10, -1.195 dB en 100x10, 1.98 dB en 100x20 y -0.22 dB en 128x16. En esta iteración, 100x10 y 128x16 son las configuraciones que mejor rendimiento ofrecen ya que requieren un menor valor de SNR, ambas obtienen un resultado muy similar. El peor resultado se obtiene al utilizar la configuración 50x10, con un resultado de entre tres y cuatro dB por encima de 100x10 o 128x16.

En cuanto a 64-QAM, se obtienen unos resultados de 8.12 dB para 50x10, 4 dB para 100x10, 6.85 dB para 100x20 y 4.15 dB para 128x16. Al igual que con 16-QAM, las configuraciones 100x10 y 128x16 son las que ofrecen un mejor resultado y con valores muy similares y, además, 50x10 sigue siendo la que peor rendimiento obtiene de las cuatro. Estos resultados se pueden apreciar en la Figura 36.



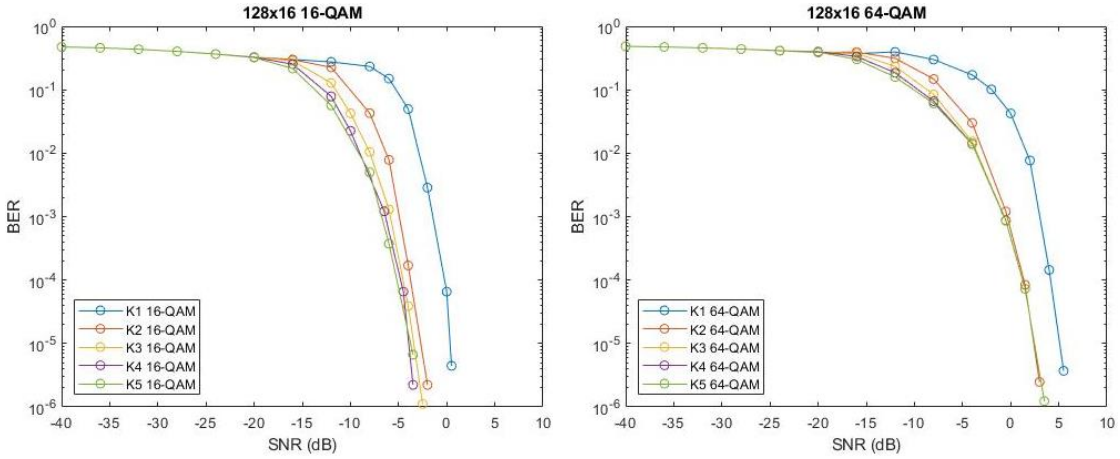


Figura 36: Rendimiento en términos de BER del algoritmo ADMIN para las diferentes configuraciones seleccionadas.

Al incrementar en uno las iteraciones del algoritmo, como se puede apreciar en la Figura 36, los resultados de todas las configuraciones mejoran tanto para 16-QAM como para 64-QAM. En 16-QAM se consigue un BER de 10^{-4} con 0.55 dB en 50x10, -3.39 dB en 100x10, -1.27 dB en 100x20 y -3.75 dB en 128x16. Las configuraciones que presentan mejores resultados son 100x10 y 128x16, con valores muy similares, siendo 128x16 la que presenta un mayor incremento respecto a la iteración anterior.

En cuanto a 64-QAM, se puede observar una mejora respecto a los resultados obtenidos con una iteración. Con dos iteraciones se obtiene 6.22 dB en 50x10, 2.2 dB en 100x10, 3.21 dB en 100x20 y 1.37 dB en 128x16 para alcanzar un BER de 10^{-4}. Aunque la configuración 128x16 es la que mejor resultado presenta, 100x20 es la que ha presentado una mayor reducción en SNR respecto a la iteración anterior.

Ejecutando el algoritmo con tres iteraciones se puede apreciar una mejora en los resultados de todas las configuraciones, sin embargo, como se puede apreciar en la Figura 36, la mejora producida es muy reducida. En 16-QAM se obtienen unos valores de 0.14 dB en 50x10, -4.03 en 100x10, -2.78 en 100x20 y -4.54 en 128x16. En todos los casos la diferencia es inferior a 1 dB.

En 64-QAM ocurre lo mismo, las diferencias en el valor de SNR son mínimas. Para alcanzar un BER de 10^{-4} se requieren 5.82 dB en 50x10, 2 dB en 100x10, 2.9 dB en 100x20 y 1.25 dB en 128x16. Como en 16-QAM, la diferencia no supera 1 dB.

De la misma forma que en la iteración anterior, al ejecutar el algoritmo con cuatro iteraciones no se obtiene una mejora significativa en los resultados. Esa variación es igual o inferior en comparación con los obtenidos con tres iteraciones. En 16-QAM se obtienen -0.14 dB en 50x10, -4.03 dB en 100x10, -3.24 dB en 100x20 y -4.8 dB en 128x16. En 50x10 el resultado obtenido es el mismo que en la iteración anterior.

Para 64-QAM, se obtienen 5.82 dB en 50x10, 1.75 dB en 100x10, 2.78 dB en 100x20 y 1.24 dB en 128x16. Los resultados presentan una variación mínima respecto a la iteración anterior y, en el caso de 50x10, no hay diferencia en el resultado.

Al ejecutar el algoritmo con 5 iteraciones se produce el mismo patrón que con tres y cuatro iteraciones, los resultados presentan una variación mínima respecto a la iteración anterior. Además, en algunas configuraciones, los resultados empeoran respecto al resultado obtenido con cuatro iteraciones. En 16-QAM se obtienen unos resultados de -0.4 dB en 50x10, -4 dB en 100x10, -3.28 dB en 100x20 y -5.18 dB en 128x16. Se puede observar que, en el caso de 100x10, el resultado obtenido empeora respecto al obtenido con cuatro iteraciones.

En 64-QAM se obtienen como resultados 6.11 dB en 50x10, 2.39 dB en 100x10, 2.65 dB en 100x20 y 1.24 dB en 128x16. En este caso, 100x20 es la única configuración que mejora el resultado en comparación con el obtenido con cuatro iteraciones. La configuración 128x16 no varía y las configuraciones 50x10 y 100x10 obtienen un resultado peor que en la iteración anterior.

A partir de los resultados obtenidos en las cinco iteraciones diferentes realizadas, se puede determinar que la configuración 128x16 es la que ofrece un mejor resultado. Aunque al ejecutar el algoritmo con una sola iteración la configuración 100x10 obtiene un mejor resultado, estos no son muy dispares entre sí y, además, en las siguientes iteraciones 128x16 mejora los resultados de 100x10.

También cabe destacar que, a partir de la tercera iteración, los resultados no mejoran en gran medida. La mejora que se produce es mínima y, en algunas configuraciones, llega hasta empeorar. Debido a esto no tendría sentido aplicar el algoritmo ADMIN con más de tres iteraciones. En la Tabla 4 se puede observar una comparativa de los resultados obtenidos en cada iteración.

		Número de iteraciones				
ADMIN		1	2	3	4	5
50x10	16-QAM	3.02	0.55	0.14	-0.14	-0.4
	64-QAM	8,12	6.22	5.82	5.82	6.11
100x10	16-QAM	-1.195	-3.39	-4.03	-4.03	-4
	64-QAM	4	2.2	2	1.75	2.39
100x20	16-QAM	1.98	-1.27	-2.78	-3.24	-3.28
	64-QAM	6.84	3.21	2.9	2.78	2.65
128x16	16-QAM	-0.22	-3.75	-4.54	-4.8	-5.18
	64-QAM	4.15	1.37	1.25	1.24	1.24

Tabla 4: Diferentes valores de SNR (dB) para los cuales se obtiene un BER igual a 10^{-4} en función de la configuración, modulación e iteración.

5.2.3 Simulaciones SOR

Antes de realizar las simulaciones con el algoritmo SOR se ha calculado el valor óptimo del parámetro de relajación ω para así obtener el mejor valor de BER en cada valor de SNR. Para ello se ha escogido un valor de SNR con y se ejecuta la simulación barriendo los valores de $\boldsymbol{0 < \omega < 2}$. El valor de $\boldsymbol{\omega}$ para el cual se obtiene el menor valor de BER será el utilizado en la simulación del cálculo del BER entre los valores dados de SNR. En la Figura 37 se puede observar el resultado obtenido para la configuración 128x16 en 16-QAM y con una iteración. En este caso se ha seleccionado $\boldsymbol{\omega = 1.1}$.

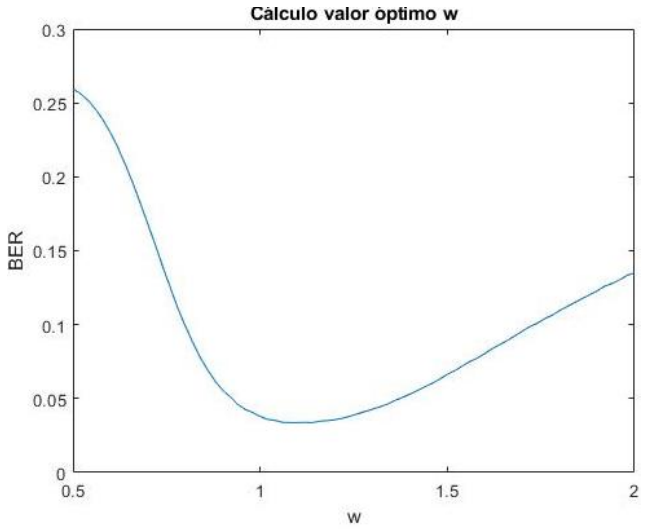
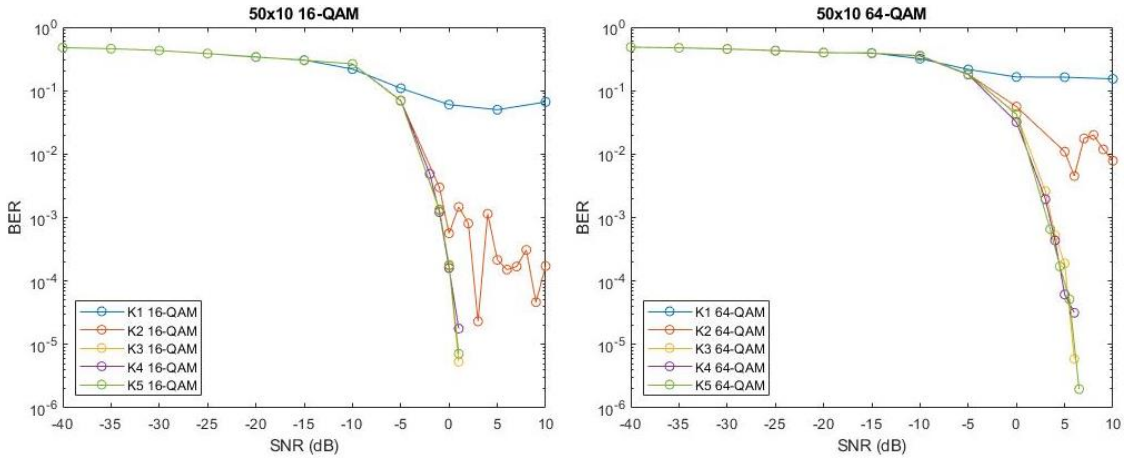


Figura 37: Cálculo parámetro w para 128x16 en 16-QAM con una iteración.

La simulación del algoritmo con una iteración muestra unos resultados de BER que no serían aptos para transmitir datos de forma fiable, de la misma forma que en el algoritmo OCD-BOX con una iteración. En todas las configuraciones, en valores de SNR superiores a 30 dB no se llega a conseguir un BER nulo, el BER tiende a mantenerse estable sin llegar a ser cero ningún momento. Debido a esto, el algoritmo SOR con una iteración no sería apto como detector en ninguna de las configuraciones evaluadas. En la Figura 38 se pueden observar los resultados obtenidos en las diferentes iteraciones.



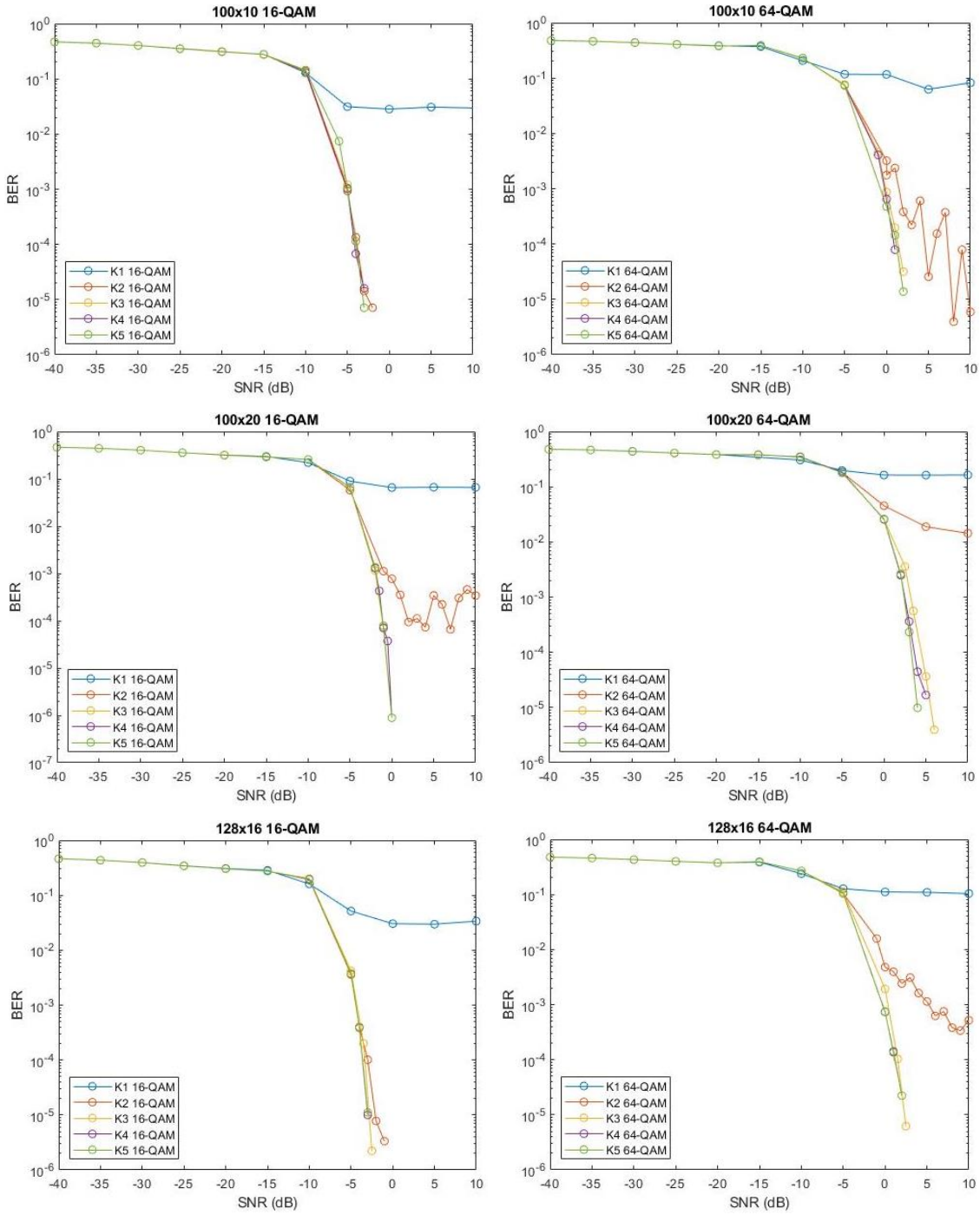


Figura 38: Rendimiento en términos de BER del algoritmo SOR para las diferentes configuraciones seleccionadas.

Al ejecutar la simulación con dos iteraciones las configuraciones 100x10 y 128x16 con modulación 16-QAM son las únicas que presentan un BER que alcanza un valor de 10^{-4}, como se puede observar en la Figura 38. Aunque la configuración 100x10 en modulación 64-QAM sí que alcanza un valor de 10^{-4}, los resultados son muy inestables y la recta muestra una tendencia a estabilizarse. En 100x10 se obtiene un BER de 10^{-4} con -3.87 dB, y en 128x16 con -3 dB.

Realizando la simulación del algoritmo con tres iteraciones, en todas las configuraciones se alcanza un BER de 10^{-4}. En la modulación 16-QAM se consigue con 0.16 dB en 50x10, -3.87 dB en 100x10, -1.13 en 100x20 y -3.35 dB en 128x16. En este caso la configuración 100x10 ofrece un mejor resultado, aun obteniendo el mismo que con la iteración anterior. Por otro lado, 50x10 es la configuración que peor resultado ofrece.

En cuanto a 64-QAM, se obtienen como resultado 5.18 dB en 50x10, 1.37 dB en 100x10, 4.45 dB en 100x20 y 1.51 dB en 128x16 para alcanzar un BER de 10^{-4}. Como en 16-QAM, la configuración 100x10 es la que mejor resultado ofrece, siendo 50x10 la peor.

Al ejecutar el algoritmo con cuatro iteraciones se obtiene en 16-QAM los siguientes resultados: 0.21 dB en 50x10, -4.14 dB en 100x10 y -1.09 dB en 100x20 y -3.62 dB en 128x16. En las configuraciones 50x10 y 100x20 se puede apreciar que ambos resultados han empeorado respecto a los obtenidos en la iteración anterior. Sin embargo, las configuraciones 100x10 y 128x16 han obtenido un resultado levemente mejor.

Por otro lado, en 64-QAM se obtiene como resultado 4.75 dB en 50x10, 0.89 dB en 100x10, 3.61 dB en 100x20 y 1.19 dB en 128x16. En este caso, las cuatro configuraciones han mejorado sus resultados respecto a la iteración anterior.

Por último, ejecutando el algoritmo con cuatro iteraciones se obtiene como resultado para 16-QAM 0.18 dB en 50x10, -3.95 dB en 100x10, -1.09 dB en 100x20 y -3.63 dB en 128x16. En este caso, las configuraciones 100x20 y 128x16 obtienen prácticamente los mismos resultados que en la iteración anterior y la configuración 100x10 presenta la mayor mejora de entre las cuatro.

En 64-QAM, se obtienen 4.94 dB en 50x10, 1.16 dB en 100x10, 3.27 dB en 100x20 y 1.17 dB en 128x16. Esto hace que las configuraciones 100x20 y 128x16 presenten una leve mejora respecto a los resultados de la iteración anterior y que las configuraciones 50x10 y 100x10 muestren un resultado peor que en la iteración anterior.

Según los resultados obtenidos de las cinco iteraciones diferentes que se han realizado, el algoritmo SOR requiere de un mínimo de tres iteraciones para que el detector tenga un rendimiento aceptable. Como se puede observar en los resultados obtenidos, las configuraciones que mejor rendimiento ofrecen con SOR son 100x10 y 128x16. Como se puede observar en los resultados de la Tabla 5, utilizar estas configuraciones con cuatro iteraciones es cuando se obtienen resultados óptimos para 100x10 y 128x16. Por lo tanto, ya que la variación en los resultados es mínima, no tiene sentido realizar cinco iteraciones con el algoritmo SOR.

		Número de iteraciones				
SOR		1	2	3	4	5
50x10	16-QAM	--	--	0.16	0.21	0.18
	64-QAM	--	--	5.18	4.75	4.94
100x10	16-QAM	--	-3.87	-3.87	-4.14	-3.95
	64-QAM	--	--	1.37	0.89	1.16
100x20	16-QAM	--	--	-1.13	-1.09	-1.09
	64-QAM	--	--	4.45	3.61	3.27
128x16	16-QAM	--	-3	-3.35	-3.62	-3.63
	64-QAM	--	--	1.51	1.19	1.17

Tabla 5: Diferentes valores de SNR (dB) para los cuales se obtiene un BER igual a 10^{-4} en función de la configuración, modulación e iteración.

5.2.4 Simulaciones SLAS

Como se ha explicado anteriormente, en SLAS no se tienen en cuenta todos los vecinos pertenecientes a un vecindario de tamaño L para actualizar el valor del coste ML. Para ello, se calcula un vecindario reducido utilizando una regla de selección: K sets más probables, selección basada en normalización o selección basada en normalización de suma de cuadrados.

Antes de realizar las diferentes simulaciones, se han comparado los resultados que se obtienen en una configuración al utilizar una regla de selección u otra. Para ello se han realizado tres simulaciones diferentes, cada una con una regla de selección, para observar si se produce una variación en el BER obtenido. La prueba se ha realizado con la configuración 100x10 y modulación 16-QAM.

Como se puede observar en la Figura 39, la selección de una u otra regla de selección tiene un impacto mínimo en el BER obtenido. Debido a esto, las diferentes simulaciones se han realizado aplicando la selección basada en normalización, con un umbral de 0.8, el mismo que se aplica en [17].

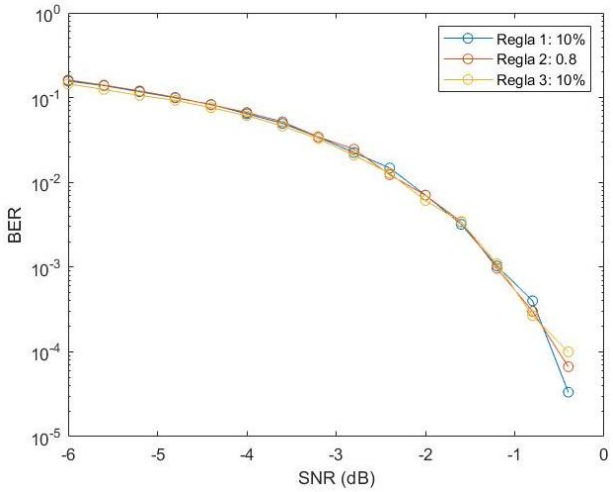


Figura 39: Resultados de la comparación entre las diferentes reglas de selección del algoritmo SLAS.

A diferencia de los algoritmos anteriores, en el caso de SLAS solo se han realizado simulaciones con tamaño de vecindario **L**=1. Esto se debe a que el tiempo requerido para realizar las diferentes simulaciones con valores de K más elevados es extremadamente alto. Como ejemplo, al ejecutar las simulaciones para **L**=2 en el servidor del GTAC, tardó dos semanas en realizar 5 iteraciones del total de 400 para cada valor de SNR que se evalúa.

En la Figura 40 se muestran las gráficas de los resultados obtenidos de las diferentes ejecuciones de las simulaciones para el cálculo del BER en cada configuración.

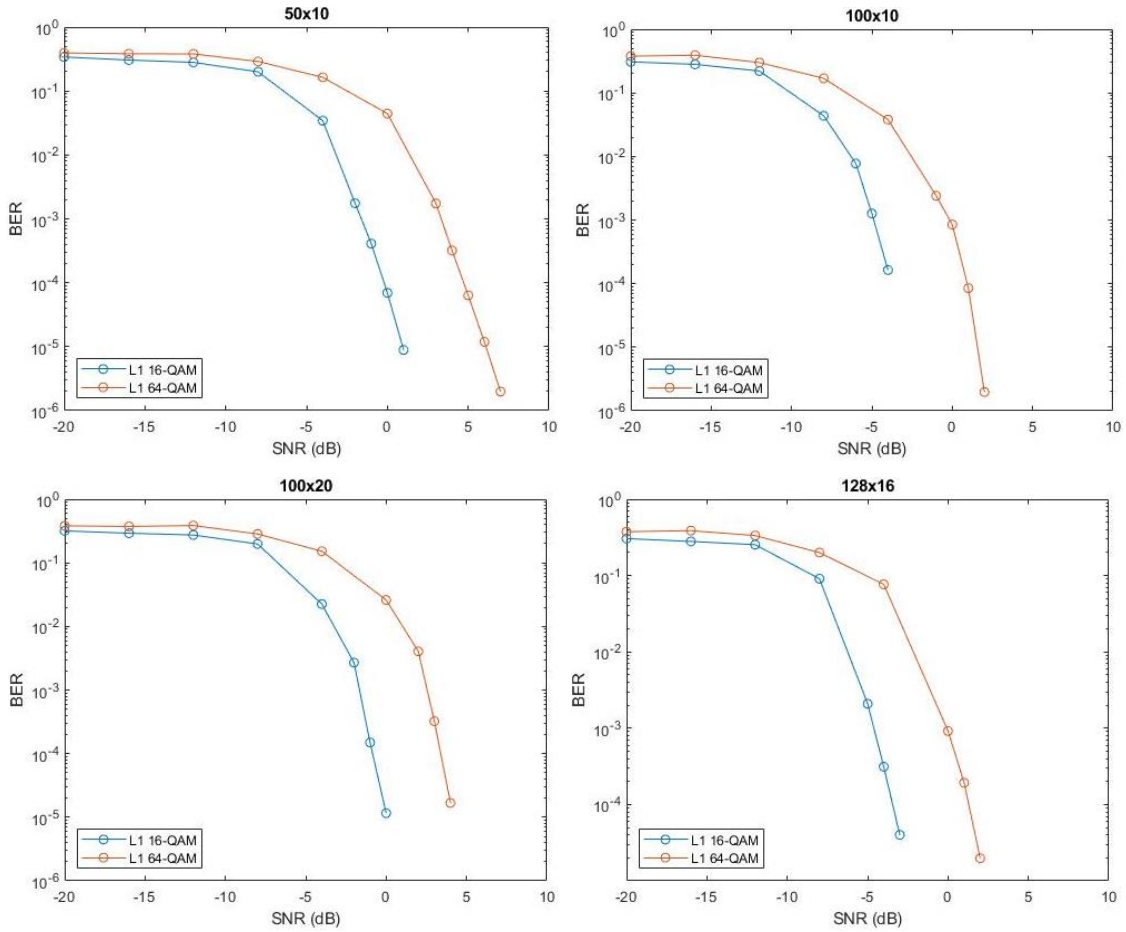


Figura 40: Rendimiento en términos de BER del algoritmo SLAS para las diferentes configuraciones seleccionadas.

Como se puede observar en las gráficas obtenidas, con un tamaño de vecindario de uno se obtienen resultados, tanto en 16-QAM como en 64-QAM, que tienden a cero en las cuatro configuraciones.

En la configuración 50x10 se obtiene que, para alcanzar un BER de 10^{-4}, se requiere un valor de SNR de -0.21 dB en 16-QAM y de 4.72 dB en 64-QAM, en 100x10 los resultados son de -4 dB en 16-QAM y 0.93 dB en 64-QAM, en 100x20 son de -0.84 dB para 16-QAM y 3.4 para 64-QAM y en 128x16 son de -3.45 dB para 16-QAM y 1.29 dB para 64-QAM.

Con estos resultados se puede apreciar que el algoritmo SLAS, con un tamaño de vecindario igual a uno, obtiene resultados similares a los algoritmos anteriormente mencionados para los que requieren 4 o 5 iteraciones. Las configuraciones que obtienen un mejor rendimiento son 128x16 para 16-QAM y 100x10 para 64-QAM. En la Tabla 6 se pueden observar los resultados obtenidos en todas las configuraciones.

Tamaño de vecindario		
SLAS		1
50x10	16-QAM	**-0.21**
	64-QAM	**4.72**
100x10	16-QAM	**-4**
	64-QAM	**0.93**
100x20	16-QAM	**-0.84**
	64-QAM	**3.4**
128x16	16-QAM	**-3.45**
	64-QAM	**1.29**

Tabla 6: Diferentes valores de SNR (dB) para los cuales se obtiene un BER igual a 10^{-4} en función de la configuración y modulación.

Con los resultados de los diferentes algoritmos de detección empleados se puede observar que, entre los detectores *BOX-constrained*, ADMIN es el que obtiene un mejor rendimiento comparado con OCD-BOX. Esto es debido a que con ADMIN se obtienen mejores resultados en iteraciones bajas ya que OCD-BOX nunca consigue alcanzar un BER igual, o inferior, a 10^{-4} con menos de 3 iteraciones. Además, ambos algoritmos obtienen resultados similares en todas las configuraciones cuando las iteraciones son mayores a 3. Es por esto que ADMIN, si solo se tiene en cuenta el BER, obtiene mejores resultados.

Comparando los cuatro algoritmos, el que ofrece un mejor resultado para 16-QAM es ADMIN con la configuración 128x16 y en 64-QAM se consigue un mejor resultado con el algoritmo SOR utilizando la configuración 100x10.

5.3 Prueba de transmisión de video

Al conocer que valores de SNR son los adecuados para cada algoritmo, configuración y modulación empleada, se ha elegido el mejor caso de cada uno de ellos para realizar una simulación del envío de un video a través del canal. Mediante esta prueba se podrá comparar la calidad de la imagen recibida sobre el mejor escenario posible de cada algoritmo.

Para la transmisión del video se ha escogido utilizar únicamente la modulación 64-QAM. Esto es debido a que con la introducción de la modulación 256-QAM, 16-QAM comienza a quedarse desfasada en su uso.

Comparando los resultados obtenidos en cada algoritmo, se ha escogido la configuración que ha presentado un menor valor de SNR para alcanzar un BER igual a 10^{-4}. En el caso de los algoritmos *BOX-constrained*, OCD-BOX y ADMIN, ambos obtienen los mejores resultados cuando se hace uso de la configuración 128x16 y en cada algoritmo el número de iteraciones utilizado es 5.

Por otro lado, en los algoritmos de detección lineal basada en métodos para la aproximación de la matriz invertida y en basados en búsqueda local, SOR y SLAS, la configuración que ha obtenido mejor resultado ha sido 100x10. En el caso de SOR se ha obtenido con un total de 4 iteraciones. En la Tabla 7 se agrupan las configuraciones elegidas junto con sus valores de SNR.

Algoritmo	Configuración	Valor de SNR (dB)
OCD-BOX	128x16	1.25
ADMIN	128x16	1.24
SOR	100x10	0.89
SLAS	100x10	0.93

Tabla 7: Diferentes configuraciones seleccionadas para simular la transmisión de video en el canal MIMO.

Para los cuatro casos se ha transmitido el mismo video. Como referencia para el transmisor de video usado en el dron se han tomado las especificaciones del transmisor digital empleado por la marca DJI. Como valores se han tomado una tasa de bits de 50 Mb/s, la máxima soportada por el transmisor, y se ha supuesto que trabaja en modo de baja latencia ya que este es el más exigente. En este modo la resolución del video es de 1440x810 y tiene una tasa de fotogramas por segundo de 120.

La transmisión del video se ha realizado fotograma a fotograma, los cuales son recibidos y escritos en un nuevo vídeo. El fotograma leído se convierte previamente a binario y, al emplear modulación 64-QAM, los bits se juntan en grupos de 6. Dado que la secuencia obtenida es de gran longitud y el tiempo requerido para realizar las operaciones con matrices es elevado, se ha optado por partir la secuencia en grupos más pequeños para reducir la carga de dichas operaciones.

Por último, para determinar la diferencia de calidad entre el video original y el recibido se ha hecho uso de la herramienta FFmpeg y de la escala de puntuación VMAF (*Video Multi-method Assessment Fusion*) [19]. VMAF ha sido desarrollado por Netflix en colaboración con *University of Southern California* y es una métrica de calidad de video que combina el modelo de visión humana junto con aprendizaje máquina para ofrecer unos resultados más acertados que otras métricas como PSNR (*Peak-Signal-to-Noise ratio*) o SSIM (*Structural Similarity Index Measure*). VMAF dispone de diferentes modelos pensados para ciertas situaciones, en este caso se ha empleado el modelo estándar adaptado para teléfonos móviles. Este modelo es el más adecuado ya que contempla una resolución de 720p, similar a la resolución empleada en la simulación.

Al aplicar el filtro libvmaf de FFmpeg sobre los videos recibidos se obtiene una puntuación de VMAF entre 20 y 100. Las puntuaciones obtenidas son las siguientes: ADMIN 96.79, OCD-BOX 96.66, SOR 80.28 y SLAS 80.93.

En la Figura 41 se puede observar la comparación entre el mismo fotograma en cada una de las simulaciones. En el caso de ADMIN y OCD-BOX se aprecia una imagen más limpia y clara, mientras que en SOR y SLAS la imagen es más difusa y cuesta más apreciar los detalles.

A)

B)

C)

D)

Figura 41: Comparación entre el fotograma 103 de los diferentes algoritmos: ADMIN (A), OCD-BOX (B), SOR (C), SLAS (D).

En el contexto de una carrera de drones lo más importante no es recibir una imagen nítida que permita observar todos los detalles de la escena, sino recibir la transmisión sin cortes para que el piloto no pierda la concentración en ningún momento evitando posibles accidentes. Además, hay que tener en cuenta que en la transmisión realizada no se ha tenido en cuenta ninguna forma de corrección de errores o procesado de video en el receptor para mejorar la calidad de la imagen recibida. Es por esto que, en base a la comparación de la calidad recibida, los cuatro algoritmos serían aptos para su uso en la transmisión de video, siendo los algoritmos ADMIN y OCD-BOX los que ofrecen una mejor calidad de imagen.

5.4 Complejidad de los algoritmos

Otro aspecto a tener en cuenta al evaluar que algoritmo es el más adecuado es el coste computacional requerido para la ejecución del propio algoritmo. Este aspecto es especialmente importante ya que determinará los requisitos hardware en la BS para mantener la latencia de la imagen en unos valores aceptables.

Para el cálculo de la complejidad se ha tenido en cuenta la cantidad de operaciones realizadas dentro de las funciones definidas para cada algoritmo. El coste de las operaciones es de un flop en sumas y multiplicaciones y, en el caso de ser complejos, el coste de las sumas es de 2 flops y las multiplicaciones de 4 flops.

En el caso del algoritmo OCD-BOX se ha tenido en cuenta las operaciones realizadas dentro de la función *BoxDetection_OCD()* definida para realizar la detección y la función *projeccion()*. En ambas funciones se trabaja con números complejos. El coste computacional obtenido varía en función de las antenas transmisoras y receptoras (N_t y N_r), longitud del mensaje recibido (data), modulación utilizada (M) y cantidad de iteraciones del detector (K). La función del coste del algoritmo se define de la siguiente forma:

$$\boldsymbol{N_t}\left[(\boldsymbol{8}(\boldsymbol{N_r}+\boldsymbol{1})+\boldsymbol{6Mdata})+\boldsymbol{K}\left(\boldsymbol{4N_r}(\boldsymbol{data}+\boldsymbol{1})+\boldsymbol{data}(\boldsymbol{2}(\boldsymbol{N_r}-\boldsymbol{1})+\boldsymbol{10})\right)\right]$$

En el caso del algoritmo OCD-BOX la cantidad de antenas transmisoras y el número de iteraciones son las variables que más peso tienen en el coste del algoritmo.

En el algoritmo ADMIN coste se define de manera similar, ya que también se hace uso de la función *projeccion()*. Sin embargo, la cantidad de operaciones realizadas es mayor en la función *BoxDetection_ADMIN()* ya que se realizan más operaciones en la parte de preprocesado, como la factorización de cholesky, lo que hace aumentar su coste computacional respecto a OCD-BOX.

En el caso de la factorización de Cholesky de la matriz **G** ($\boldsymbol{H^H H + \beta I}$), el coste se define de la siguiente manera:

$$N_t\left[\frac{5}{2}N_t(N_t-1)+2\right]+5\left[\sum_{x=1}^{N_t}(x-1)(x-2)\frac{1}{2}\right]+6$$

La factorización de Cholesky consiste en sumatorios que recorren todas las posiciones de la matriz, por lo tanto, el tamaño de la matriz (que en este caso será cuadrada) determina el coste de la factorización.

Por otro lado, el coste de la detección realizada por el algoritmo, cálculo de la matriz de filtrado MMSE y la detección *hard* se define de la siguiente manera:

$$K\left[2data\left(N_t^2+N_t(N_t-1)\right)+20N_t data\right]+2N_t[2N_r+(N_r-1)+2]+3Mdata$$

En este caso la cantidad de antenas transmisoras y el número de iteraciones a realizar son las variables que más peso tienen en el cálculo del coste.

En cuanto al algoritmo SOR, la función *sorMethod()* trabaja con números reales, por lo tanto el coste de las operaciones se reduce a la mitad. Sin embargo, esto no afecta al coste total del algoritmo ya que la cantidad de operaciones a realizar se duplica.

Previo a la detección, se realiza el cálculo de la matriz G y de la solución inicial. En este caso si se tiene en cuenta el coste de la solución inicial ya que el valor inicial es diferente de cero. El coste se define como el coste del preprocesado, el bucle de detección y la detección *hard*:

$$K\left[2\left(1+(2N_t)^2\right)+4N_t data(4N_t-1)+2N_t data\right]+2N_t data\left(2\sqrt{M}+(4N_r-1)\right)+(2N_t)^2 4N_r$$

Las principales variables que afectan al coste total del algoritmo son la cantidad de iteraciones del bucle de detección y la cantidad de antenas transmisoras.

Por último, el algoritmo SLAS, al igual que SOR, trabaja con números reales. Para el cálculo del coste se ha tenido en cuenta la función *SLAS()* y la función *calc_NL()* donde se realiza el cálculo de todas las combinaciones de los vecindarios a calcular.

A diferencia de los algoritmos anteriores, SLAS no se ejecuta con un número determinado de iteraciones. El algoritmo determina el tamaño de búsqueda del vecindario en función del coste ML calculado. Sin embargo, en una iteración dada, si el coste calculado es inferior al coste previo el tamaño del vecindario volverá a ser uno. Debido al comportamiento del algoritmo el coste no se podrá calcular, si no estimar. El coste del preprocesado, incluyendo solución inicial, matriz G, matriz de filtrado MMSE y coste de la detección *hard* se define por la siguiente ecuación:

$$(2N_t)^2 4N_r+2N_r\left((2N_t)^2+2N_t(2N_t-1)\right)+2N_t data\left(2N_r+(2N_r-1)\right)+4N_t data\sqrt{M}$$

El coste del cálculo de las diferentes combinaciones de todos los tamaños de vecindario (siendo *MaxStage* el máximo) se ha definido como el coste del sumatorio del cálculo de todos los tamaños de vecindario más el coste de la asignación del valor y el cambio de posición. El coste se define por la siguiente ecuación.

$$\sum_{L=1}^{MaxStage} 2\frac{2N_t!}{L!\,(2N_t-L)!}$$

En cuanto al coste del bucle de detección, la suma del coste de las diferentes operaciones se define con la siguiente ecuación:

$$I\left[\frac{2N_t!}{MaxStage!\,(2N_t - MaxStage)!} + 2N_r(8N_t + 1) + N_r(6 + 8N_t) + 2(2N_r - 1)\right]$$

En este caso no se ha tenido en cuenta la cantidad de iteraciones (I) realizadas en el bucle ya que no están definidas. Para poder realizar una comparación entre los costes de los algoritmos se ha seleccionado el valor medio de iteraciones realizado por valor de SNR en el cálculo del BER. Para 0.93 dB se tendrían $I = 2$.

Realizando el cálculo del coste para cada algoritmo (utilizando las configuraciones con mejor rendimiento) variando el valor de *data* en todos los algoritmos, se obtiene que el algoritmo con un menor coste es ADMIN y el que tiene un mayor coste es SOR. En la Figura 42 se puede observar el aumento del coste en función de la longitud del mensaje recibido.

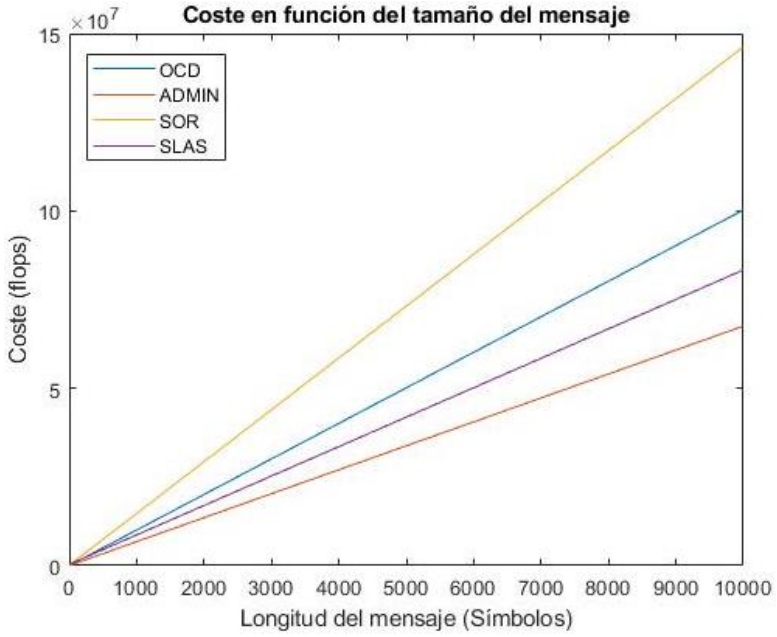


Figura 42: Comparación de coste computacional entre algoritmos en condiciones de mejor rendimiento.

5.5 Conclusiones de las pruebas realizadas

Con los resultados obtenidos del estudio del BER, la simulación de la transmisión de vídeo y el coste computacional se puede determinar que el algoritmo que ofrece un mejor rendimiento es ADMIN. Aunque SOR tiene un menor valor de SNR en su mejor configuración, la diferencia con el valor de SNR de ADMIN en su mejor configuración es muy pequeña. Además, ADMIN presenta un resultado mucho mejor al comparar las imágenes recibidas

Capítulo 6. Conclusiones y Trabajo Futuro

En el desarrollo del trabajo se ha estudiado el uso de una BS en el entorno de una carrera de drones para la transmisión de la señal de video en downlink mediante técnicas de mMIMO. En concreto, el interés del trabajo se ha centrado en el estudio de diferentes algoritmos de detección con el objetivo de conocer cual sería el más adecuado para dicho escenario, evaluando la SNR, la calidad de imagen percibida y el coste computacional.

En cuanto a los valores de SNR, los cuatro algoritmos evaluados han presentado resultados similares para alcanzar un BER 10^{-4}. En este caso la SNR obtenida no serviría como un factor determinante para descartar uno de los algoritmos, de hecho, es más útil haber observado que dos de ellos (OCD-BOX y ADMIN) alcanzan este valor con una configuración de antenas de 128x16. Este dato es importante ya que, como se ha comentado al principio del trabajo, una de las principales desventajas del sistema de transmisión analógico utilizado actualmente en carreras de drones es la imposibilidad de superar más de 8 transmisiones simultaneas. De esta manera, con los resultados obtenidos, sería posible realizar una carrera de drones con el doble de participantes manteniendo una tasa de BER aceptable.

Por otro lado, como se ha podido comprobar con las simulaciones de transmisión de video, la calidad de imagen obtenida con los cuatro algoritmos sería aceptable en el contexto de una carrera de drones. Como se ha comentado anteriormente, la calidad de la imagen no es un factor determinante para saber si es o no apta para su uso en una carrera, siempre que se encuentre en un rango aceptable. Según los resultados obtenidos tras aplicar el filtro VMAF, los algoritmos que han aplicado una detección *Box-Constrained* son los que mejor calidad de imagen han obtenido.

Por último, un factor de gran importancia a tener en cuenta para conocer que algoritmo ofrece un mejor rendimiento es el coste computacional que requiere el propio proceso de detección. La principal razón de su importancia es el impacto que tiene en el coste económico del hardware de la BS requerida para la recepción de la señal. Como se ha visto, ADMIN sería el algoritmo que presenta un menor coste y SOR presentaría el mayor coste de los cuatro.

Con las diferentes pruebas realizadas se ha llegado a la conclusión que el algoritmo ADMIN es el más adecuado para implementar una BS con técnicas mMIMO.

6.1 Líneas de Trabajo Futuro

Aunque en el trabajo realizado se ha completado el objetivo inicial del proyecto, esto no quiere decir que no se pueda seguir explorando y encontrar mejores soluciones.

Uno de los principales puntos en los que se tendría que continuar trabajando es en la búsqueda de **nuevos algoritmos en los que realizar pruebas** para conocer si se obtienen mejores resultados que los obtenidos en este trabajo. Como resultado del trabajo realizado se ha llegado a la conclusión que los algoritmos que emplean una detección *Box-Constrained* ofrecen un mejor rendimiento, por lo tanto, sería interesante buscar más algoritmos similares para comparar los resultados.

Con el resultado obtenido, sería posible realizar una carrera con un total de 16 participantes de manera simultánea, o incluso 8 participantes con 2 antenas en cada dron. Debido a esto, otro punto a explorar es la **comparación de más configuraciones** en las que se tenga un mayor número de antenas transmisoras y receptoras con el objetivo de aumentar la cantidad de participantes en las carreras y la calidad de la imagen recibida.

Como continuación del trabajo realizado sería interesante probar si el uso de un BS con técnicas mMIMO es posible aplicarlo en el caso de que las cámaras en el dron transmitieran una **imagen 360º**. Hoy en día, las aplicaciones de realidad virtual han adquirido una mayor popularidad ya que las gafas que se encuentran en el mercado empiezan a tener precios más asequibles. Es por esto que la transmisión del contenido de la carrera en 360º, además de ofrecer un mayor campo de visión al piloto, permitirá al usuario disfrutar de la experiencia con una mayor inmersión.

Capítulo 7. Bibliografía

[1] https://es.statista.com/estudio/40579/la-industria-mundial-de-drones/

[2] https://patents.google.com/patent/US10032125B1/en?oq=US10032125B1

[3] https://patents.google.com/patent/US9459620

[4] https://www.fomento.gob.es/NR/rdonlyres/7B974E30-2BD2-46E5-BEE5-26E00851A455/148411/PlanEstrategicoDrones.pdf

[5] https://www.3gpp.org/

[6] https://www.fatshark.com/product/attitude-v6/

[7] https://www.dji.com/es/dji-fpv/specs

[8] https://ieeexplore.ieee.org/Xplore/home.jsp

[9] https://www.ffmpeg.org/

[10] E. G. Larsson, "MIMO Detection Methods: How They Work [Lecture Notes]," in IEEE Signal Processing Magazine, vol. 26, no. 3, pp. 91-95, May 2009, doi: 10.1109/MSP.2009.932126.

[11] M. A. Albreem, M. Juntti and S. Shahabuddin, "Massive MIMO Detection Techniques: A Survey," in IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3109-3132, Fourthquarter 2019, doi: 10.1109/COMST.2019.2935810.

[12] M. Wu, C. Dick, J. R. Cavallaro and C. Studer, "High-Throughput Data Detection for Massive MU-MIMO-OFDM Using Coordinate Descent," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 63, no. 12, pp. 2357-2367, Dec. 2016, doi: 10.1109/TCSI.2016.2611645.

[13] S. Shahabuddin, M. Juntti and C. Studer, "ADMM-based infinity norm detection for large MU-MIMO: Algorithm and VLSI architecture," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050311.

[14] X. Gao, L. Dai, Y. Hu, Z. Wang and Z. Wang, "Matrix inversion-less signal detection using SOR method for uplink large-scale MIMO systems," 2014 IEEE Global Communications Conference, 2014, pp. 3291-3295, doi: 10.1109/GLOCOM.2014.7037314.

[15] Hadjidimos, A. (2000). Successive overrelaxation (SOR) and related methods. Journal of Computational and Applied Mathematics, 123(1–2), 177–199. https://doi.org/10.1016/S0377-0427(00)00403-9

[16] K. Sah and A. K. Chaturvedi, "Sequential and Global Likelihood Ascent Search-Based Detection in Large MIMO Systems," in IEEE Transactions on Communications, vol. 66, no. 2, pp. 713-725, Feb. 2018, doi: 10.1109/TCOMM.2017.2761383.

[17] https://www.openstreetmap.org/#map=18/39.47446/-0.35788

[18] Shahriar Shahabuddin (2021). Massive MIMO Detection (https://www.mathworks.com/matlabcentral/fileexchange/68767-massive-mimo-detection), MATLAB Central File Exchange.

[19] VMAF: The Journey Continues (https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12)

Capítulo 8. Anexo

8.1 Código Implementado

8.1.1 Simulación BER

```
%% Datos
aTx = 20;                          % antenas transmisoras (drones)
aRx = 100;                         % antenas receptoras (BS)
NFFT = 5120;                       % Tamaño fft
NdataPilotOFDM = 3010;             % Subportadoras datos + piloto
NGuard = 2110;                     % Subportadoras guarda
SNR1 = -40:1:1;                   % Ruido
SNR2 = 1:0.5:6;                   % Ruido
SNR  = [SNR1 SNR2];
M = 64;                            % Modulacion M-QAM
c = qammod(0:M-1,M,'gray');        % Constelación de la modulación M-QAM
Es = mean(abs(c).^2);              % Energia por símbolo
k = log2(M);                       % Bits/symbolo
Modpilot = 2;                      % Modulacion pilotos (BPSK)
Ncp = 64;                          % Longitud prefijo ciclico

% Parametros simulacion
Nsim1 = 300;                       % simulación/SNR1(i)
Nsim2 = 600;                       % simulación/SNR2(i)
K = 2;                             % Iteraciones máximas
gamma = 0.5;                       % Step size
SepPilot = 15;                     % Separación de puntos entre pilotos
NbitsTx = 50000;                   % Numero de bits a transmitir por antena

NpilotOFDM = floor(NdataPilotOFDM/SepPilot);
NdataOFDM = NdataPilotOFDM-NpilotOFDM;
symOffset = Ncp;

%% Generate transmitted data
Nbits = NdataOFDM*k;
NsymOFDM = ceil(NbitsTx/Nbits);
NbitsOFDM = Nbits*NsymOFDM;
dataTx = randi([0 M-1],NdataOFDM,NsymOFDM,aTx);
symTx = qammod(dataTx, M,'gray');

%% Generate pilots
pilotTx = randi([0 Modpilot-1], NpilotOFDM*NsymOFDM*aTx,1);
symPilot = pskmod(pilotTx, Modpilot, 0, 'gray');
symPilot = reshape(symPilot,NpilotOFDM,NsymOFDM,aTx);

%% Generate OFDM Signal
pilotidx = [linspace((NGuard/2)+1,NFFT/2,NpilotOFDM/2)
linspace((NFFT/2)+2,NFFT-(NGuard/2)-1,NpilotOFDM/2)]';
nullidx = [1:(NGuard/2) (NFFT/2)+1 NdataPilotOFDM+(NGuard/2)+2:NFFT]';
Tx = ofdmmod(symTx,NFFT,Ncp,nullidx,round(pilotidx),symPilot).';
```

```
%% Simulations
vBER = zeros(length(SNR),Nsim2);
time = zeros(length(SNR),Nsim2);

Nsim = Nsim1;
f = fopen('res.txt','w');
t = fopen('tiempos.txt','w');
for i = 1:length(SNR)

    if i > length(SNR1)
       Nsim = Nsim2;
    end

    for j = 1:Nsim

        % Rayleigh distributed channel response matrix
        H = complex(randn(aRx, aTx),randn(aRx, aTx))*sqrt(0.5);

        % AWGN Channel
        AveragePower = mean(abs(Tx).^2);
        awgnchan = comm.AWGNChannel('NoiseMethod',...
            'Signal to noise ratio (SNR)', 'SNR', SNR(i),...
            'SignalPower', AveragePower);
        MIMORx = awgnchan(H*Tx);

        % Separate data and pilot
        [RxData,RxPilot] =
ofdmdemod(MIMORx.',NFFT,Ncp,symOffset,nullidx,round(pilotidx));
        RxData = reshape(RxData,NdataOFDM*NsymOFDM,aRx).';

        % ADMIN detection
        No = (Es*aTx)/(10.^(SNR(i)./10));
        tic;
        Rx = BoxDetection_ADMIN(RxData,H,No,Es,c,K,gamma);
        time(i,j) = toc;
        % Demodulation
        dataRx = qamdemod(Rx,M,'gray');
        [ ~, BER] = biterr(de2bi(reshape(dataTx,NdataOFDM*NsymOFDM,aTx).',k),
de2bi(dataRx,k));

        % Store results
        vBER(i,j) = BER;

        fprintf('SNR %i: %i\n',SNR(i),j);
    end

    BER_medio = mean(vBER(i,Nsim).');
    time_medio = mean(time(i,Nsim).');
    fprintf(f,'%4df \n',BER_medio);
    fprintf(t,'%4df \n',time_medio);
end

fclose(f);
fclose(t);
```

8.1.2 BoxDetection_ADMIN

```
function x = BoxDetection_ADMIN(y,H,No,Es,c,K,g)
%Aplicación del algoritmo de detección ADMIN
%ADMM-based infinity-norm (ADMIN) realiza box-constrained equalization
%   y : señal recibida (aRx x LenFrame)
%   H : matriz del canal MIMO (aRx x aTx)
%   No : densidad espectral de potencia del ruido
%   Es : Energia por símbolo
%   K : Numero de iteraciones
%   c : Símbolos de la constelación utilizada

    %Preprocesado
    U = size(H,2); % Numero de transmisores (usuarios con 1 antena)
    beta = No/Es;
    G = H'*H + beta * eye(U);
    L = chol(G,'lower');
    m = max(real(c));

    %Inicialización
    z = zeros(U,length(y));
    lambda = zeros(U,length(y));

    %Soft Detection
    yMF = H' * y;

    for i=1:K
       x = L'\(L\(yMF + beta*(z-lambda)));
       zz = projeccion(x + lambda, m);
       lambda = lambda - g*(zz - x);
       z = zz;
    end

    %Hard detection

    for i=1:size(x,1)
        for j=1:size(x,2)
            if(~ismember(x(i,j),c))
                [~,I] = min(abs(x(i,j) - c));
                x(i,j) = c(I);
            end
        end
    end
end
```

8.1.3 BoxDetection_OCD

```
function  res = BoxDetection_OCD(y,H,c,K)
%Aplicación del algoritmo de detección OCD
%   Optimazed Coordinate Descent (OCD) para modo BOX-constrained
%   y : Señal recibida (aRx x lenFrame)
%   H : Matriz de canal MIMO (aRx x aTx)
%   c : Símbolos de la constelación utilizada
%   K : Número de iteraciones

    %Inicializacion
    U = size(H,2);  % Numero de transmisores (usuarios con 1 antena)
    z0 = zeros(U,size(y,2));
    z1 = zeros(U,size(y,2));
    zvar = zeros(U,size(y,2));
    m = max(real(c));

    %Proprocesado
    d = zeros(U,1); % (regularized) inverse squared column norms of H
    p = zeros(U,1); % regularized gains (siempre va a ser p(u)=1)
    for i=1:U
        d(i) = 1/(norm(H(:,i),2)^2);
        p(i) = d(i) * norm(H(:,i),2)^(2);
    end

    %Equalizacion
    for k=1:K
       for u=1:U
           z1(u,:) = projeccion(d(u)*(H(:,u)'*y) + p(u)*z0(u,:),m);
           zvar(u,:) = z1(u,:) - z0(u,:);
           y = y - H(:,u) * zvar(u,:);
           z0 = z1;
       end
    end

    %Hard detection

    for i=1:size(z1,1)
        for j=1:size(z1,2)
            if(~ismember(z1(i,j),c))
                [~,I] = min(abs(z1(i,j) - c));
                z1(i,j) = c(I);
            end
        end
    end

    %Resultado
    res = z1;
end
```

8.1.4 Projeccion

```
function res = projeccion(w,c)
%La salida es igual a la entrada si es un punto que se encuentra dentro
%del set de la constelación. Si no lo está, la salida es el valor más próximo
%a la entrada que se encuentra dentro del conjunto de símbolos de la
%constelación.
%   w : matriz de símbolos de entrada en el Rx
%   c : máximo valor de constelación utilizada

    res = zeros(size(w));

    for i=1:size(w,1)
        for j=1:size(w,2)

            res(i,j) = w(i,j);

            if real(w(i,j)) > c
                res(i,j) = c*sign(real(w(i,j))) + imag(w(i,j))*sqrt(-1);
            end

            if imag(w(i,j)) > c
                res(i,j) = real(w(i,j)) + c*sign(imag(w(i,j)))*sqrt(-1);
            end
        end
    end
end
```

8.1.5 sorMethod

```
function res = sorMethod_real(y,H,beta,K,w,c)
%Aplicación del algoritmo de detección SOR
%   Successive Over Relaxation (SOR) Method
%   y : Señal recibida (aRx x lenFrame)
%   H : Matriz de canal MIMO (aRx x aTx)
%   beta: valor para generar la matriz G
%   K : Número de iteraciones
%   w : Parametro de relajación
%   c : Símbolos de la constelación utilizada

    % Preprocesado
    U = size(H,2);  % Numero de antenas transmisoras (usuarios con 1 antena)
    Bs = size(H,1); % Número de antenas receptoras en estación base

    % Se trabaja con numeros reales
    y2 = zeros(2*Bs,size(y,2));
    y2(1:1:Bs,:) = real(y);
    y2(Bs+1:1:end,:) = imag(y);
    H2 = [real(H) -imag(H); imag(H) real(H)];

    G = H2'*H2 + beta * eye(2*U);
    symb = unique(real(c));
```

```
    %Inicializacion
    yMF = H2' * y2;
    x = zeros(2*U,size(y2,2));     %solucion inicial
    w = 1/w;
    D = diag(diag(G));
    L = tril(G,-1);
    Up = triu(G,1);

    %Equalizacion
    for i=1:K
        x = (w.*D + L)\(yMF + ((w-1).*D - Up)*x);
    end

    %Hard-output
    for i=1:size(x,1)
        for j=1:size(x,2)
            if(~ismember(x(i,j),symb))
                [~,I] = min(abs(x(i,j) - symb));
                x(i,j) = symb(I);
            end
        end
    end

    % Cambio de reales a complejos
    res = complex(x(1:1:U,:),x(U+1:1:2*U,:));
end
```

8.1.6 SLAS

```
function [res, iter] = SLAS2(y,H,c,maxStage,beta,idx)
%Detector SLAS (Sequential Likelihood Ascent Search)
%   y: [U x LenFrame]
%   H: matriz de canal mMIMO
%   c: Símbolos de constelación QAM utilizada
%   maxStage: tamaño de vecindario L máximo hasta el que se realiza la
%   búsqueda
%   beta: valor para generar la matriz G
%   idx: Regla de selección para las uk

    % Preprocesado
    U = size(H,2);  % Numero de antenas transmisoras (usuarios con 1 antena)
    Bs = size(H,1); % Número de antenas receptoras en estación base

    % Se trabaja con numeros reales
    y2 = zeros(2*Bs,size(y,2));
    y2(1:1:Bs,:) = real(y);
    y2(Bs+1:1:end,:) = imag(y);
    H2 = [real(H) -imag(H); imag(H) real(H)];

    G = H2.'*H2 + beta * eye(U*2);
    A_MMSE = G\H2.';
    xr = A_MMSE * y2; % Solución inicial MMSE

    dmin = min(abs(c(2:end) - c(1)));
    symb = unique(real(c));
```

```
    % Redondeo de la solución inicial a valores de la constelación M-QAM
    for i=1:size(xr,1)
        for j=1:size(xr,2)
            if(~ismember(xr(i,j),symb))
                [~,I] = min(abs(xr(i,j) - symb));
                xr(i,j) = symb(I);
            end
        end
    end

    % Inicialización
    iter = 0;
    L = ones(1,size(y2,2));
    Laux = ones(1,size(y2,2));

    for i=1:size(y2,2)
        costNext(i) = norm(y2(:,i)-H2*xr(:,i),2)^2;
    end
    costPre = ones(1,size(y2,2)) * Inf;
    xupd = zeros(size(xr));
    Ik = calc_NL4(U,maxStage);

    % Equalización

    % Mientras haya alguna L <= maxStage
    while sum(Laux) ~= 0
%% Determinar los sets de indices a actualizar

        % Generar todas las combinaciones posibles de L indices y
        % guardarlas como Ik => 1:factorial(2*U)/(factorial(L)*factorial(2*U-
L))
        Lres = find(Laux);
        disp(num2str(length(Lres)));
        NHmax = factorial(2*U)/(factorial(max(L(Lres)))*factorial(2*U-
max(L(Lres))));
        uk = zeros(1,round(NHmax),length(L));

        % Se genera la uk correspondiente a cada combinación Ik generada
        for z=1:length(Lres)
            for i=1:factorial(2*U)/(factorial(L(Lres(z)))*factorial(2*U-
L(Lres(z))))

                pos = find(Ik(:,i,L(Lres(z))));
                uk(1,i,Lres(z)) = sum((((y2(:,Lres(z))-
H2*xr(:,Lres(z))).'*H2(:,pos))/norm(H2(:,pos),2)).^2);

            end
        end

        %% Aplica la regla de decisión en uk y selecciona unos cuantos Ik

        % 1) K most likely sets selection:

        if idx == 1
            RNs = false(2*U,ceil(0.1*(NHmax)), length(L));
            for j=1:length(Lres)
```

```
                val = factorial(2*U)/(factorial(L(Lres(j)))*factorial(2*U-
L(Lres(j))));
                [~,m] = maxk(uk(1,:,Lres(j)),floor(0.1*(val)));
                RNs(:,1:length(m),Lres(j)) = Ik(:,m,L(Lres(j)));      %
Vecindarios reducidos seleccionados de cada L

            end
        end

        % 2) Normalization based selection

        if idx == 2
            RNs = zeros(2*U,round(NHmax),length(L));
            for j=1:length(Lres)
                ukMax = max(uk(1,:,Lres(j)));
                ukNorm = uk(1,:,Lres(j))./ukMax;
                pos = find(ukNorm>=0.8);
                RNs(:,pos,Lres(j)) = Ik(:,pos,L(Lres(j))); % Vecindarios
reducidos seleccionados
            end
        end

        % 3) Sum Square Normalization Based Selection !!![REVISAR]!!!

        if idx == 3
            RNs = zeros(2*U,NHmax,length(L));
            for j=1:length(Lres)
                den = sqrt(sum(abs(uk(1,:,Lres(j))).^2));
                res = uk(1,:,Lres(j))/den;
                pos = find(res > 0.05);
                RNs(:,pos,Lres(j)) = Ik(:,pos,L(Lres(j))); % Vecindarios
reducidos seleccionados
            end
        end

        costPre = costNext;
        xtemp = xr;

%% Actualizando los sets de indices seleccionados
        for z=1:length(Lres)
            for j=1:size(RNs,2)

                aux = find(RNs(:,j,Lres(z)));

                if isempty(aux)
                    break;
                end

                HH = H2(:,aux);
                er = y2(:,Lres(z))-H2*xr(:,Lres(z));
                n = ceil((((HH.' * HH).^-1) * HH.' * er)/dmin);
                xupd(:,Lres(z)) = xr(:,Lres(z));
                xupd(aux,Lres(z)) = xr(aux,Lres(z)) + n*dmin;

                %% Actualizando el mejor vector disponible
                costTemp = norm(y2(:,Lres(z)) - H2*xupd(:,Lres(z)),2)^2;
                if costTemp < costNext(Lres(z))
                    xtemp(:,Lres(z)) = xupd(:,Lres(z));
```

```
                    costNext(Lres(z)) = costTemp;
                end
            end
        end
%% Determinación del tamaño del vecindario para la búsqueda

        for i=1:length(Lres)
            if(costNext(Lres(i)) < costPre(Lres(i)))
                L(Lres(i)) = 1;
            else
                L(Lres(i)) = L(Lres(i)) + 1;
                if L(Lres(i)) > maxStage
                    Laux(Lres(i)) = 0;
                end
            end
        end

        xr = xtemp;
        iter = iter + 1;
    end

    % Redondeo a valores de la constelación
    for i=1:size(xr,1)
        for j=1:size(xr,2)
            if(~ismember(xr(i,j),symb))
                [~,I] = min(abs(xr(i,j) - symb));
                xr(i,j) = symb(I);
            end
        end
    end

    % Cambio de reales a complejos
    res = complex(xr(1:1:U,:),xr(U+1:1:2*U,:));

end
```

8.1.7 calc_NL

```
function Ik = calc_NL4(U,maxStage)
%CALC_NL4 Devuelve Los vecindarios de cada valor de L
%   maxStage: máximo tamaño de vecindario
%   U: Número de antenas transmisoras

    % Diferentes combinaciones de vecinos con L símbolos de diferencia
    comb = factorial(2*U)./(factorial(1:maxStage).*factorial(2*U-
(1:maxStage)));
    Ik = false(2*U,round(max(comb)),length(comb));
    vec = false(2*U,1);

    % Se recorren los valores de L que se han de actualizar
    for z=1:length(comb)

        vec(1:z) = 1;
        Ik(:,1,z) = vec;
```

```
        % Se recorren todas las combinaciones posibles del binomio (2Nt
maxStage(i))
        for i=2:comb(z)

            %Posiciones actuales de los 1 en el vector
            pos = find(vec);

            % Si la posición más baja no ha llegado al final
            if max(pos) < 2*U

               vec(max(pos)) = 0;
               vec(max(pos)+1) = 1;
            end

            % Si la posición más baja ha llegado al final y L>1
            if (max(pos) == 2*U) && length(pos) > 1

               % Se busca el siguiente valor que se puede bajar 1 posición
               for e=length(pos)-1:-1:1

                   if vec(pos(e)+1) ~= 1

                       vec(pos(e)) = 0;
                       vec(pos(e)+1) = 1;
                       pos(e) = pos(e) + 1;
                       for h=e+1:length(pos)

                           vec(pos(h)) = 0;
                           vec(pos(e)+(h-e)) = 1;
                       end
                       % Una vez movido un valor, se termina el bucle
                       break;
                   end
               end
            end

            % Se incluye el vector en el vecindario de la L correspondiente
            Ik(:,i,z) = vec;
        end

        % Se devuelve a ceros para el siguiente valor de L
        vec = zeros(2*U,1);
    end
```