UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

DISEÑO DE UN SISTEMA OCC PARA DISPOSITIVOS WEARABLE

Titulación: Grado en Ingeniería en Tecnologías de la Telecomunicación
Autor: D. Antonio Ramón Mederos Barrera
Tutores: Dr. Rafael Pérez Jiménez
D. Cristo Manuel Jurado Verdú

Fecha: Diciembre de 2019

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

DISEÑO DE UN SISTEMA OCC PARA DISPOSITIVOS WEARABLE

HOJA DE EVALUACIÓN

Calificación: _____

Presidente

Fdo.:

Vocal **Secretario/a**

Fdo.: **Fdo.:**

Fecha: Diciembre de 2019

"Aprender a convertir la razón en una pasión."

Spinoza - Irvin Yalom.

Dedicado a mi abuela.

Agradecimientos

A mis padres, mis hermanas y familia por ser el pilar fundamental de mi vida. El amor incondicional, el apoyo que dan sin esperar nada a cambio, en todas las circunstancias de la vida, hacen a cualquier persona una excepción.

A mis tutores Rafael y Cristo, por ofrecerme la oportunidad de explorar nuevas ideas, y no poner límites a la razón. La comprensión, la paciencia y la motivación han sido los motivos por los que he crecido como persona, fuera de lo académico.

A mis amigos por estar siempre en todo momento, por la mano amiga y por la confianza. Por la superación, el crecimiento personal y las noches de estudio y filosofía.

A mis compañeros por hacer el laboratorio un lugar cálido y acogedor.

A todos ustedes, muchas gracias.

Índice general

Índice de figuras

Índice de tablas

Índice de códigos

Parte I

Memoria

Capítulo 1

Introducción

1.1. Antecedentes

En la actualidad, el aumento de los dispositivos conectados a Internet, promovido por la creciente expansión del Internet of Things (IoT), genera una gran demanda de comunicaciones inalámbricas, lo que provoca la saturación del espectro radioeléctrico. Este hecho impulsa la búsqueda de soluciones alternativas que alivien la carga de los canales de radio convencionales al tiempo que se aseguren velocidades de transmisión altas. Aparecen así las Optical Wireless Communications (OWC), que se posicionan como una tecnología alternativa en el sector de las comunicaciones. En concreto, dentro del campo las Comunicaciones Ópticas, se sitúa la rama de las Comunicaciones por Luz Visible o tecnología Visible Light Communications (VLC) [5], la cual utiliza el amplio espectro no regulado de la luz visible. El ánimo principal de esta tecnología consiste en utilizar la actual infraestructura de luminarias Light-Emitting Diode (LED) para la transmisión de información, al tiempo que preservan su función principal como sistema de iluminación. En lo que respecta a los sistemas receptores, se utilizan comúnmente uno o varios fotorreceptores independientes.

El interés por esta tecnología es tal que el Institute of Electrical and Electronics Engineers (IEEE) tiene un grupo asignado, el IEEE 802.15.7, para la regulación y normalización de estos sistemas, así como sus aplicaciones [6].

Sin embargo, todavía no existen soluciones comerciales que integren esta reciente tecnología, lo cual ralentiza considerablemente su entrada en el mercado, así como, la búsqueda de las aplicaciones finales realmente útiles para el usuario. Por esto, nace la

idea de utilizar los sensores de captación de imagen o cámaras como sistemas receptores. Estos elementos sí son bastante comunes dentro de la sociedad actual, ya que son muchos los dispositivos personales, tales como ordenadores portátiles y teléfonos móviles los que integran ya una cámara de altas prestaciones.

A esta rama tecnológica se le conoce como Optical Camera Communications (OCC) o Comunicaciones Ópticas basadas en Cámara [7][8][9]. Los sistemas OCC comparten con cualquier sistema de comunicación los bloques generales de transmisor, canal y receptor. En la Figura 1.2 se muestra el esquema general de un sistema OCC. El

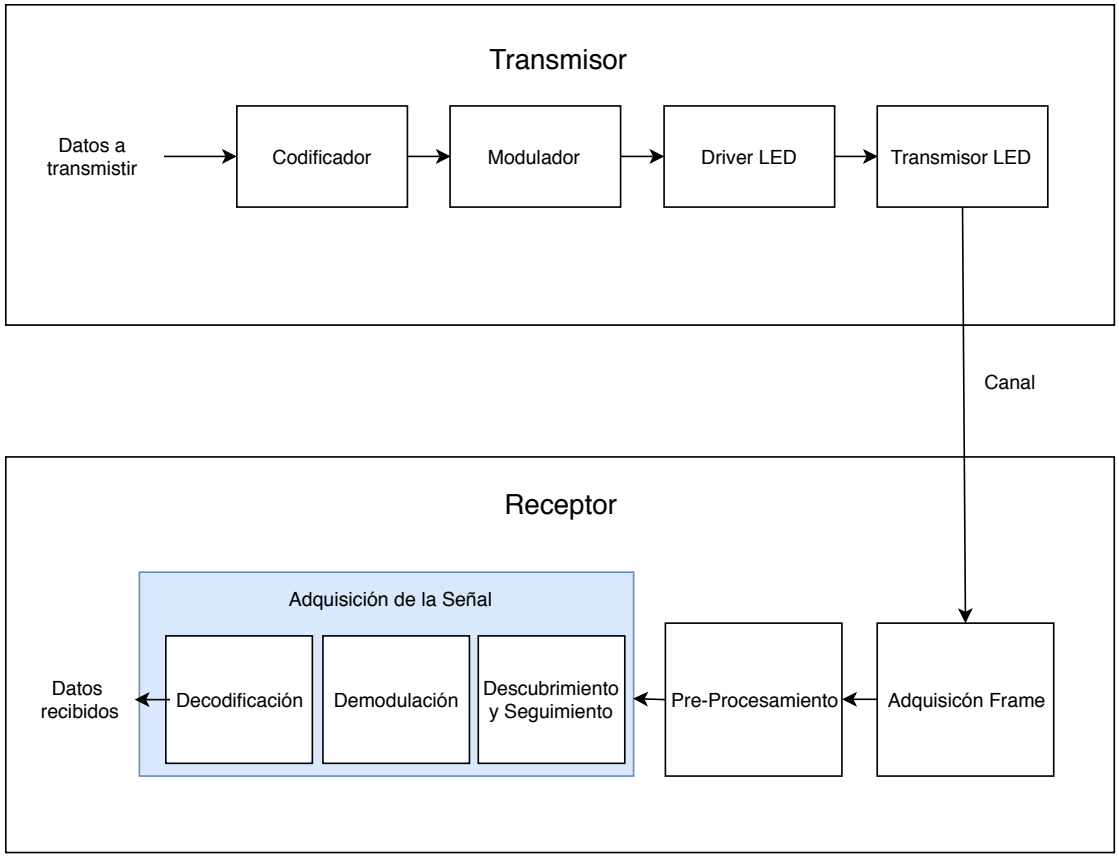

Figura 1.1: Esquema general de un sistema OCC.

transmisor se puede subdividir en:

- **Modulador**: codifica y modula la señal para ser transmitida por el canal. Dos de las modulaciones más utilizadas son la On-Off Keying (OOK) y la Undersampled Phase Shift OOK (UPSOOK) [10]. La OOK consiste en transmitir estados de encendido "ON" y apagado "OFF", que representan la información lógica. Sin embargo, la tasa de captura limitada de las cámaras (de unos pocos *frame*s por segundo) obliga a que las lámparas conmuten a una velocidad relativamente baja

(atendiendo al criterio de Nyquist), lo cual provoca un parpadeo apreciable. Para hacer frente a esta situación, aparece la modulación UPSOOK la cual modula una portadora que tiene una frecuencia considerablemente mayor a la tasa de captura de la cámara.

- **Driver LEDs**: alimenta y controla los LED, protegiéndolos para que estos no se dañen, y trabajen en una región lineal.
- **Transmisor LEDs**: genera la señal óptica a partir de la señal eléctrica. Compuesta por LEDs normalmente Red-Green-Blue (RGB).

El receptor se puede dividir en subsistemas de:

- **Obtención de *frame***: recoge la potencia óptica incidente sobre la superficie de los fotorreceptores del sensor de la cámara y la transforma en valores digitales para cada píxel y canal RGB conformando una imagen.
- **Pre-procesamiento**: preprocesa la imagen, preparándola para ser tratada mediante técnicas de procesamiento digital para la extracción de la señal de información.
- **Adquisición de la señal**: obtiene los datos recibidos. Se compone, a su vez, de varios bloques o etapas: el descubrimiento y seguimiento de la fuente transmisora, la extracción de la señal, la demodulación y la posterior decodificación y corrección de errores.

Dado su principio de funcionamiento, la transmisión utilizando la luz, los sistemas de comunicación óptica no generan interferencia radioeléctrica con los equipos médicos, al contrario que las comunicaciones vía radio. Esto hace que los sistemas OCC tengan una aplicación útil en los hospitales y salas de urgencias para la monitorización de pacientes [11].

El objetivo del presente proyecto consiste en utilizar esta reciente tecnología para adaptarla a los dispositivos *wearables* o portables. Estos son dispositivos portables que se suelen insertar en prendas o accesorios. En la Figura **??** se puede observar un ejemplo de dispositivo *wearable* combinado con un sistema transmisor OCC. En este campo (los sistemas *wereables*), la tecnología OCC presenta una serie de retos y dificultades asociados a la movilidad y la probabilidad de oclusión de estos dispositivos.

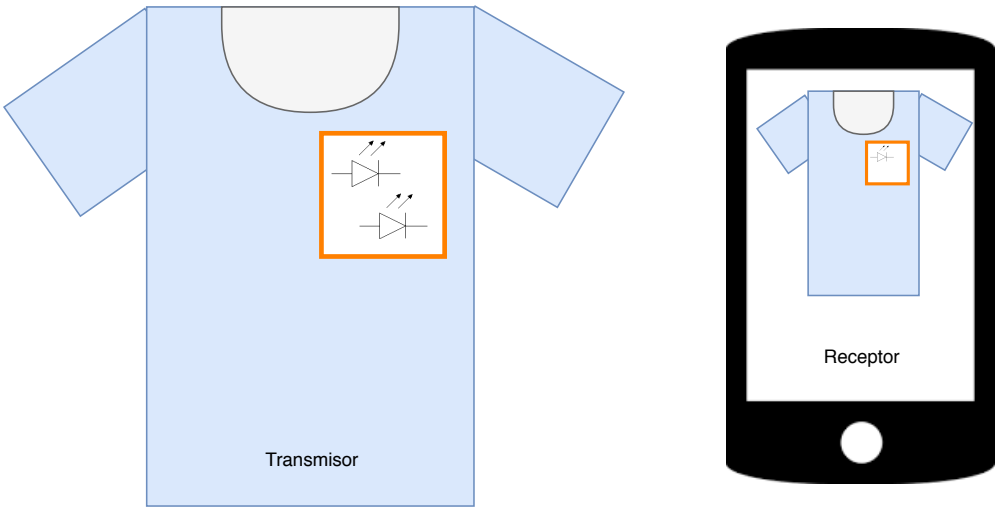

Figura 1.2: Sistema OCC en dispositivos wearable.

Las actuales investigaciones en OCC tratan comunicaciones estáticas, en las que transmisor y receptos no se mueven, o bien, la posición relativa entre ambos sistemas no varía mucho en el tiempo. En este sentido, la integración de estos sistemas OCC en dispositivos wearables tienen que operar bajo condiciones opuestas, implementando sistemas de descubrimiento y seguimiento, para que, una vez descubierta la fuente, el sistema receptor sea capaz de seguirla a través del tiempo antes de adquirir la información.

Entre las soluciones propuestas para el descubrimiento de la fuente se propone utilizar el envío periódico de una señal temporal conocida previamente por el receptor, llamada cabecera. Esta técnica es común para el descubrimiento de faros en redes vehiculares [8-10]. En caso de perder la fuente en la imagen, es necesario rastrear dicha señal a lo largo del tiempo, lo que reduce considerablemente la tasa de transmisión efectiva.Además, dado que la señal debe ser captada en distintos *frames* (muestreo temporal) la fuente debe estar perfectamente localizada entre todos los *frames*, por lo que el transmisor debe moverse lentamente, o de moverse con una cierta velocidad, es necesario corregir los efectos de este movimiento.

Como alternativa se propone aprovechar la diversidad espacial que ofrecen las cámaras y configurar el transmisor espacialmente para ser detectado en un único *frame*.

Por último, la probabilidad de oclusión de la fuente transmisora ha sido estudiada en [12] y como mecanismo de protección es común utilizar redundancia en la transmisión, lo cual reduce la tasa de transmisión efectiva.

1.2. Objetivos

El objeto del presente trabajo es el estudio y el diseño del sistema transmisor-receptor OCC en el marco de los dispositivos wearables. Respecto a la fuente transmisora, se ha de diseñar un sistema wearable capaz de transmitir la señal óptica y ha de ser, a su vez, portable y ligero. Respecto al receptor, se han de diseñar las etapas de descubrimiento y seguimiento de la fuente. Este objetivo se divide en los siguientes objetivos fundamentales definidos para este Trabajo Fin de Grado:

O1. Estudiar y documentar el estado de técnica en el campo de las comunicaciones OCC.

O2. Analizar los problemas de los dispositivos wearables y los problemas de los sistemas OCC con estos dispositivos.

O3. Diseñar el sistema de transmisión wearable.

O4. Diseñar el sistema receptor.

O5. Plantear las métricas de evaluación, el diseño del entorno experimental y la verificación final.

1.3. Peticionario

Actúa como peticionario del presente Trabajo de Fin de Grado (TFG) la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE) de la Universidad de Las Palmas de Gran Canaria (ULPGC) como requisito indispensable para la obtención del título del Grado en Ingeniería en Tecnologías de la Telecomunicación, tras haber superado con éxito las asignaturas especificadas en el plan de estudios.

1.4. Organización de la memoria

La presente memoria está estructurada de forma similar a la realización de las tareas durante el desarrollo del trabajo. El capítulo 2 documenta el marco teórico de los sistemas de comuniciones ópticas por cámara. El capítulo 3 expone la arquitectura

del sistema para la resolución del problema que se plantea, donde se definen y detallan los distintos subsistemas. El capítulo 4 detalla la implementación de los subsistemas diseñados en este trabajo. El capítulo 5 recoge la evaluación realizada del sistema donde se determinan las métricas utilizadas y el diseño e implementación del entorno de evaluación. En el capítulo 6 se exponen y discuten los distintos resultados obtenidos en la evaluación. Finalmente, en el capítulo 7 se presentan las conclusiones y se proponen líneas futuras de investigación.

Capítulo 2

Marco teórico

Los sistemas de comunicaciones ópticos son los sistemas que utilizan la luz para las comunicaciones. Como todos los sistemas de comunicación, una de las formas de clasificarlos es mediante el medio de transmisión por el cual viaja la información. Por tanto, se dividen en sistemas con medios guiados y medios no guiados. Los medios guiados o alámbricos son aquellos donde la luz queda confinada en una fibra óptica. Los medios no guiados o inalámbricos son aquellos donde se transmite la información a través del espacio libre.

Dentro de las comunicaciones ópticas no guiadas se encuentran las Comunicaciones por Luz Visible o VLC que hacen uso del espectro visible, relativo al rango de longitudes de onda comprendido entre 375 y 780 nm. En VLC se hace uso de transmisores LED para transmitir luz a partir de una señal eléctrica modulada en corriente (modulación por intensidad, IM), y fotorreceptores, elementos opto-eléctricos, para convertir la potencia luminosa recibida en su superficie en una señal eléctrica (detección directa, DD).

VLC viene promovido por el creciente aumento de dispositivos conectados gracias a la creciente aparición de nuevas aplicacionesIoT. Esto provoca que el espectro radioeléctrico se encuentre actualmente saturado. Por esto, en la búsqueda de soluciones, se desarrolla VLC como una tecnología de soporte a las actuales redes de radio desplegadas.

La principal motivación de los sistemas VLC es utilizar la infraestructura de iluminación ya existente como base para las comunicaciones. Dado que las infraestructuras de iluminación afectan a los seres humanos, se deben tener en cuenta una serie de

restricciones a la hora de diseñar estos sistemas VLC:

- El color de la lámpara debe ser adecuado a la aplicación. En caso de usar lámparas en hospitales, estas deben tener un color de temperatura frío; y en el caso de una vivienda, un color de temperatura más cálido.
- La intensidad luminosa debe comprenderse entre unos márgenes considerados para la seguridad de los seres humanos (en términos de fatiga visual y mareos).
- El *flickering* o parpadeo no debe ser apreciado por los seres humanos. Este fenómeno es provocado por el cambio de intensidad en la conmutación durante el envío de datos (1 y 0 binarios).

Pese a las ventajas que esta tecnología presenta, aún se encuentra en una fase inicial de desarrollo y su despliegue en terminales finales de usuario está ralentizado dado que la mayoría de los dispositivos electrónicos no integran este tipo de fotorreceptores. Sin embargo, poseen cámaras integradas que pueden ser utilizadas como receptores ópticos. Del uso de las cámaras para la comunicación nace las Comunicaciones Ópticas por Cámara u OCC.

Los sistemas OCC comparten las mismas características que los sistemas VLC, añadiendo la principal ventaja del acercamiento de las comunicaciones ópticas no guiadas al público general. No obstante, las comunicaciones OCC presentan los siguientes retos:

- **Reducida tasa binaria** debido a las tasas de captura (en Frames Per Second (FPS)) limitadas de las cámaras de uso común. Como solución a este reto, se puede hacer uso de la diversidad espacial que ofrecen los sensores de imagen para enviar, en el mismo *frame*, distinta información en diferentes puntos de la imagen.
- **Localización de la fuente** debido a la diversidad espacial es necesario localizar la fuente en la imagen. Ello conlleva el uso de técnicas de localización que tengan en cuenta el movimiento de la fuente.
- **Bajo poder computacional** de los terminales finales que integran cámara usados como dispositivos receptores. Por tanto, es necesario diseñar métodos de localización con una arquitectura sencilla, y esquemas de modulación y codificación con un coste eficiente.

2.1. Cámaras

Las cámaras son dispositivos que convierten la luz que proviene de diversos caminos en una señal eléctrica. La información obtenida, por tanto, presenta diversidad espacial.

El principio de las cámaras es sencillo, se tiene un sensor de imagen que consiste en una matriz de fotorreceptores, sintonizados a diferentes longitudes de ondas a través de filtros ópticos. Estos filtros siguen una configuración típica conocida como filtro de Bayer.

Este filtro consiste en una matriz de fotorreceptores donde el 50 % de ellos son verde (mayor transmitividad para el rango de longitudes de onda pertenecientes al verde), el 25 % rojo y el 25 % restante azul. Esto se debe a que el ojo humano es más sensible al verde que al resto de colores. Las filas impares consisten en una combinación de fotorreceptores verde-rojo-vede-rojo, mientras que las filas pares consisten en una combinación de azul-verde-azul-verde. La combinación de dos fotorreceptores verde-rojo en una fila par más dos fotorreceptores azul-verde en una fila impar da como resultado un valor analógico de un píxel.

Junto al filtro de Bayer se encuentra el sensor que obtiene una señal digital de la potencia óptica que alcanza a los fotorreceptores. Para implementar este dispositivo, se usan comúnmente las tecnologías CCD y CMOS. Finalmente, un Analog to Digital Converter (ADC) convierte las señales analógicas en señales digitales, donde el número de bits de cuantificación nos determina el nivel máximo de salida. En la práctica, el más usado es de 8 bits, lo que implica que el valor de cada píxel se encuentre entre 0 y 255 $(2^8 - 1)$.

Los sensores de imagen basados en CCD o Dispositivo de Carga Acoplada se basan en dispositivos de carga acopladas (Charged Couple Devices o simplemente CCD) para transportar la información de cada píxel al ADC. Los dispositivos CCD están dispuestos en cada fila de fotorreceptores y se basan en la transferencia de energía por condensadores acoplados, por lo que un dato de un fotorreceptor alejado del ADC circula por los condensadores contiguos, gracias al acoplamiento, hasta llegar al ADC. En la Figura se muestra la configuración típica de un sensor de imagen CCD conocida como *Interline Transfer CCD*.

Este funcionamiento hace que la lectura de la imagen sea lenta, dado que se debe

esperar a leer toda una fila para leer la siguiente (y lo mismo con las columnas) de forma secuencial. Dada esta configuración, estos sistemas suelen adquirir la imagen operando en modo *Global Shutter*. En este modo el sensor de imagen se expone al mismo tiempo para, posteriormente, leer el valor obtenido en cada píxel de forma secuencial. Por tanto, dada las limitaciones en la rapidez de la lectura, la tasa de captura de imágenes se reduce considerablemente.

Por otro lado, los sensores de imagen basados en tecnología CMOS o Semiconductor Complementario de Óxido Metálico se basan en la obtención de la información de cada píxel mediante el direccionamiento. Además, para cada píxel se implementa un amplificador de bajo ruido. Este tipo de sensor se implementa con tecnología de Silicio, por lo que el coste de fabricación es mucho menor a los sensores basados por CCD. Por esto, existe un mayor número de dispositivos electrónicos con sensores CMOS.

En la tecnología CMOS, la lectura de un fotorreceptor se realiza mediante la línea de selección de fila y la línea de selección de columna. Cuando una fila y una columna determinada están seleccionadas, el dato almacenado se obtiene y se entrega a un amplificador operacional y, posteriormente a su amplificación, al ADC. Para poder leer la información de todos los fotorreceptores y conformar la imagen, en primer lugar, se activa una fila y se leen los datos almacenados en las columnas. Posteriormente, se van activando y leyendo las filas siguientes. Sin embargo, antes de terminar de leer una fila, se activa la siguiente, por lo que la lectura se realiza fila a fila y de forma solapada. Este modo de adquisición es el más común y se denomina *Rolling Shutter*. No obstante, este modo puede provocar dos efectos de distorsión: el *skew* o distorsión oblicua donde la imagen aparece doblada en diagonal, y el *aliasing* espacial, producida por escanear objetos que tienen una velocidad superior o similar a la velocidad con que se leen las filas, lo que produce que partes del objeto se repetitan en distintos fragmentos de la imagen o que algunas sean omitidas. Este efecto es un recurso aprovechado en [13][14][15] para aumentar la tasa de transferencia efectiva.

Otro parámetro importante en el funcionamiento en la captura de imágenes es el tiempo de exposición, tiempo en el cual se integra la potencia óptica para cada fila de píxeles. En el caso de tener un tiempo de exposición bajo, se exponen los píxeles durante un menor tiempo, integrando menos energía luminosa, lo que produce que objetos con poca luminosidad no aparezcan en detalle. En caso contrario, si se tiene

a) Aliasing espacial b) Scew o distorsión oblicua

Figura 2.1: Efectos provocados por el Rolling Shutter.

un tiempo de exposición alto, los pixeles se exponen durante un mayor tiempo, y los objetos menos luminosos se podrán observar con más detalle, sin embargo, los objetos más luminosos tienden a la saturación. Por otro lado, en todos los sensores de imagen es posible variar la ganancia de amplificación lo que permite adaptar la sensibilidad del receptor, a la par que afecta inversamente a la generación de ruido en la imagen (Figura 2.2).

a) Tiempo exposición medio a) Tiempo exposición bajo

Figura 2.2: Efecto del tiempo de exposición en las imágenes.

2.2. Las cámaras en la tecnología OCC

Las cámaras en la tecnología OCC sirven como receptores ópticos con diversidad espacial, esto es, la posibilidad de contener información de uno o más objetos distintos localizados en diferentes posiciones de la imagen, comportándose como sistemas VLC que operan en modo SIMO (múltiples salidas para entrada única) o MIMO (múltiples salidas para múltiples entradas).

Las cámaras como receptores ópticos presentan un parámetro determinante en el rendimiento de los sistemas OCC que es la tasa de captura de imágenes por segundo (FPS). Dado que el valor de FPS de una cámara media está entre los 30 y los 120 *frames* por segundo, las tasas binarias de los sistemas OCC se ve intrínsecamente limitada. Según el criterio de Nyquist, si se tiene una frecuencia de muestro determinada, los datos se deben transmitir como máximo la mitad de esta frecuencia para evitar el *aliasing* temporal, esto es, entre 15 y 60 Hz, respectivamente. Por tanto, las tasas binarias alcanzables son relativamente bajas.

Este problema se puede solucionar de dos formas distintas. La primera consiste en aumentar los FPS del sistema. Sin embargo, en muchos casos esto no es posible ya que vienen predefinidos por fábrica. La segunda opción, consiste en aprovechar la diversidad espacial. Como se indica al comienzo, las cámaras tienen la propiedad de detectar objetos distintos en posiciones diferentes, por lo que, enviando datos de diversas fuentes, se pueden obtener tasas de transmisión mucho mayores. Esto se conoce como modulación espacial. Por ejemplo, para un LED con una cámara a 60 FPS, la tasa binaria máxima es de 30 bit/s; si se posicionan dos LED en puntos distintos de la imagen que compartan la transmisión de datos, la tasa binaria se ve duplicada. El número de LEDs usados en esta solución nos define el factor de escala de la capacidad de la comunicación.

Sin embargo, para poder obtener los diferentes datos de los transmisores de uno o varios LEDs, es preciso localizar la/s fuente/s transmisora/s dentro de la imagen.

2.3. Detección de fuentes transmisoras en OCC

Para detectar la/s fuente/s transmisoras y aprovechar la diversidad espacial en la comunicación, se puede hacer uso de dos técnicas de localización o detección: la temporal o la espacial.

La detección temporal consiste en el envío de un código conocido por el receptor, llamado baliza. Para el envío de la baliza se interrumpe la transmisión de datos. Esto produce que la tasa de transferencia efectiva se vea afectada. Además, se precisa de perfecta localización de la fuente entre los distintos frames, lo que obliga a que la fuente se encuentre estática de forma relativa al receptor, o utilizar complejos algoritmos para la corrección del movimiento de la fuente. Esta detección temporal se ha usado de redes vehiculares en [16] [17] .

Por otra parte, la detección espacial hace uso de los algoritmos de detección de objetos para descubrir la posición del transmisor en el *frame* basándose en las características de la fuente (forma, color, intensidad luminosa, etc.). Esto permite que la fuente pueda transmitir constantemente sin interrumpir la transmisión de datos para el envío de la baliza. Esta técnica utiliza detección *in frame*. Este tipo de detección también se ha usado en redes vehiculares [18].

En definitiva, el problema que presenta la detección temporal es la reducción de la capacidad o la máxima transferencia de datos de la comunicación, el cual no afecta en la detección especial. Sin embargo, existen problemas relacionados con el coste computacional y la sensibilidad y precisión de los algoritmos utilizados para la detección de objetos.

Los algoritmos de detección actuales para la localización de objetos suponen un costo computacional elevado. Este costo, más el uso de dispositivos electrónicos en las comunicaciones OCC con *hardware* limitado, hace que la detección sea imposible de realizar en algunos casos en tiempo real. Por esto la detección se divide en dos fases, el descubrimiento y el seguimiento.

En el descubrimiento, la fuente es localizada con la información de una única imagen. Este subsistema va a ser siempre necesario, pero tiene un coste mayor. Por esto, mientras menos se realice, mejor rendimiento total presentará el sistema.

En el seguimiento, con la información de los *frames* anteriores, se obtiene una región

de la imagen o Region Of Interest (ROI) donde es muy probable que se encuentre la fuente. Ya que en la etapa de seguimiento se analiza una región menor, el tiempo de ejecución de estos sistemas es menor.

2.4. Localización de objetos en imágenes

La visión por ordenador o visión artificial es la disciplina científica que estudia los distintos métodos que permiten analizar imágenes. Unas de las funciones que engloba son la clasificación de imágenes y la detección de objetos. La principal diferencia es que, en la clasificación, solo se indica el contenido de la imagen en general, mientras que, en la detección de objetos, se intenta obtener la región mínima donde se encuentra un objeto deseado dentro de la imagen. Esta última función, la de localizar, es de vital importancia en los sistemas OCC ya que, como se indica anteriormente, nos permite realizar comunicaciones con diversidad espacial.

La tendencia actual de los algoritmos de detección de objetos en imágenes está enfocada hacia la inteligencia artificial. La clasificación se puede realizar mediante una red neuronal artificial (ANN o NN). Sin embargo, estas redes son muy costosas computacionalmente para el tratamiento de imágenes, tanto el entrenamiento como la predicción, dadas las dimensiones de la imagen. Para solucionar este problema surgen las redes neuronales convolucionales (CNN). Estas redes implementan dos capas extras que se suelen utilizar antes de una red de clasificación: la capa de convolución, que consiste en un arreglo de *kernels* o filtros que permiten extraer las características de la fuente (a esta capa le suele seguir otra que aplica una transformación no lineal), y una capa *polling* (Ej: *max-polling*) que permite diezmar las muestras de las características, reduciendo la dimensionalidad de trabajo.

Sin embargo, el análisis de imágenes con distintas escalas implica tener modelos distintos para cada dimensión o aplicar mecanismos de adaptación que reducen o no aportan información (diezmado e interpolación). Para solventar este problema se implementa la técnica de ventanas deslizantes, donde se generan distintas ventanas de distintos tamaños y se clasifica su contenido. Las ventanas se van desplazando a lo largo de la imagen. El desplazamiento con la clasificación nos permite saber dónde se sitúa el objeto, por lo que se entiende que esta técnica se encuentra en localización

de objetos. No obstante, generar distintas ventanas y desplazarlas supone un coste computacional alto.

Como alternativa se proponen algoritmos de generación de *proposals*, los cuales a partir de una imagen obtienen ROIs donde es muy probable que se encuentre un objeto indeterminado. A esta región se conoce como región propuesto o *proposal*.

Los algoritmos como R-CNN y YOLO trabajan sobre este fundamento para localizar objetos en una imagen.

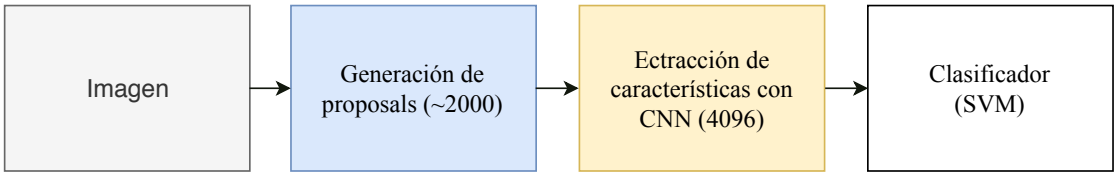

Figura 2.3: Esquema general del algoritmo R-CNN.

En la Figura 2.3 se muestra el esquema del algoritmo R-CNN propuesto por Ross Girshick et al [19]. En el algoritmo propuesto, se generan 2000 regiones de interés, lo que supuso un avance considerable frente a la técnica de ventanas deslizantes. Posteriormente se combinan regiones parecidas para reducir el número de *proposals* y se alimenta a una CNN que extrae las características (4096) de la imagen. Finalmente se clasifican los valores de las características mediante un Support Vector Machine (SVM) para determinar de qué tipo de objeto se trata. Una vez clasificada, se determina que la región estudiada posee un objeto de una clase dada.

Sin embargo, este algoritmo aún solventando los problemas anteriores, también presenta un tiempo de ejecución relativamente grande (aunque menor). Además, no se puede implementar en tiempo real ya que de media tarda 47 segundos debido a que la extracción de características se realiza para cada *proposal*.

Como alternativa al R-CNN, el mismo autor diseñó el algoritmo Fast R-CNN [20] basado en su predecesor. En este algoritmo, se genera un mapa de características a partir de la imagen total y se generan los *proposals* a partir de la imagen. Gracias a una capa de de agrupación, se fusiona la información de los *proposals* y el mapa de características, y con la información resultante, se alimenta una red *fully connected* (una ANN con todas las capas neuronas conectadas una a una).

Ya que no se generan 2000 *proposals*, Fast R-CNN es significativamente más rápido que R-CNN, pasando de tiempos de entrenamiento de 82 a 8.75 horas, y tiempos de

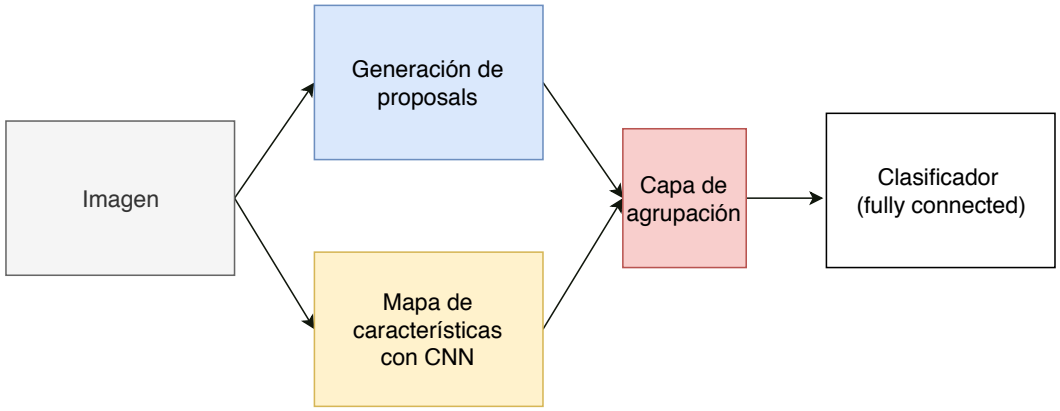

Figura 2.4: Esquema general del algoritmo Fast R-CNN.

test de 49 a 2.3 segundos. Sin embargo, los tiempos de ejecución no son lo suficientemente bajos como para ser utilizados en las comunicaciones ópticas, ya que precisan de tiempos de localización menores.

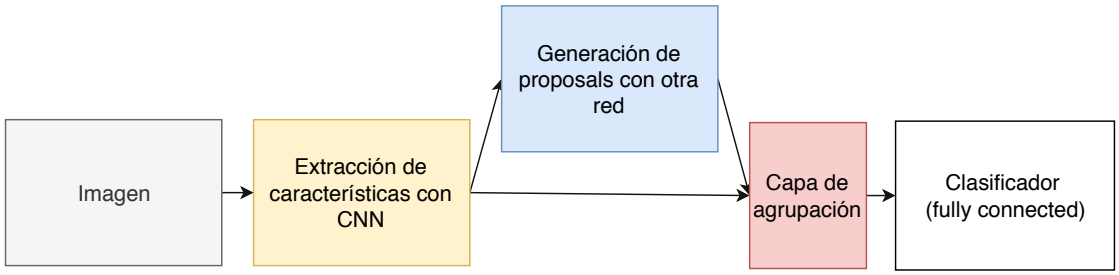

Figura 2.5: Esquema general del algoritmo Faster R-CNN.

El tiempo de ejecución elevado en R-CNN y Fast R-CNN es debido al algoritmo utilizado en la generación de *proposals*, el Selective Search, que es costoso y lento. Por esto, a Shaoqing Ren et al [21], se les ocurrió una arquitectura que elimina la necesidad de este algoritmo, conocido como Faster R-CNN. Para ello, en primer lugar, se genera el mapa de características de toda la imagen, y sobre este mapa se generan los *proposals*. La generación de *proposals* en este caso se realiza con otra red neuronal separada. Finalmente, se utiliza una capa de agrupación para fusionar la información de los *proposals* y el mapa de características. La información resultante se alimenta a una *fully connected*.

Este algoritmo de localización presenta un tiempo de test de 0.5 segundos, (recordar que el tiempo en Fast R-CNN y R-CNN es de 2.3 y 49 segundos respectivamente), por lo que este algoritmo se puede aplicar en aplicaciones de tiempo real.

Sin embargo, aun teniendo un tiempo de test bajo, Faster R-CNN posee una arquitectura costosa debido al uso de una CNN para extracción de características, una

red para la generación de *proposals* y una *fully connected* para la clasificación. Lo cual implica costes de computación y almacenamiento, por lo que supone que no se pueda usar en los dispositivos electrónicos con *hardware* limitado.

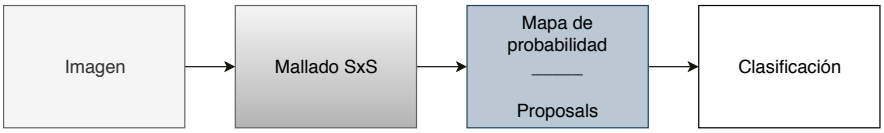

Figura 2.6: Esquema general del algoritmo YOLO.

Por otro lado, se encuentra el algoritmo You Only Look Once (YOLO) [22] que difiere considerablemente de las arquitecturas anteriores. En el caso de este algoritmo no se hace uso de ventanas deslizantes ni la generación de textitproposals, sino que toma en cuenta la imagen completa a la hora de realizar predicciones. Este algoritmo tiene en cuenta la imagen completa también durante el entrenamiento y el textittest.

El funcionamiento de YOLO se basa en dividir la imagen en una matriz de SxS celdas. Una única CNN se encarga de predecir N ROIs junto con la probabilidad de que esa región pertenezca a una clase. Finalmente, se aplica un umbral de probabilidad de clase y se seleccionan aquellas ROI que tengan mayor probabilidad.

Este algoritmo permite tasas rápidas de procesamiento de 45 FPS. Sin embargo, en la detección de objetos de tamaño reducido es necesario aumentar el número de celdas, esto implica, aumentar los recursos de computación, y aun así la sensibilidad de detección seguiría siendo bastante baja. Este inconveniente es crucial para los sistemas OCC, ya que los LEDs de los transmisores se pueden considerar como fuentes puntuales dentro de la imagen. Por este motivo, la eficiacia del YOLO en OCC es reducida.

Finalmente, cabe destacar que en todos los algoritmos presentados presentan los siguientes bloques comunes (Figura 2.7): extracción de características, generación de *proposals* y clasificación. En base a esto, si se desea un algoritmo optimizado para los sistemas OCC, se deben atacar estos tres subsistemas para poder diseñar una arquitectura eficiente.

2.5. Algoritmo de generación de proposals

En los algoritmos de localización o descubrimiento citados en la sección anterior, se comenta el uso del Selective Search como generador de *proposals*.

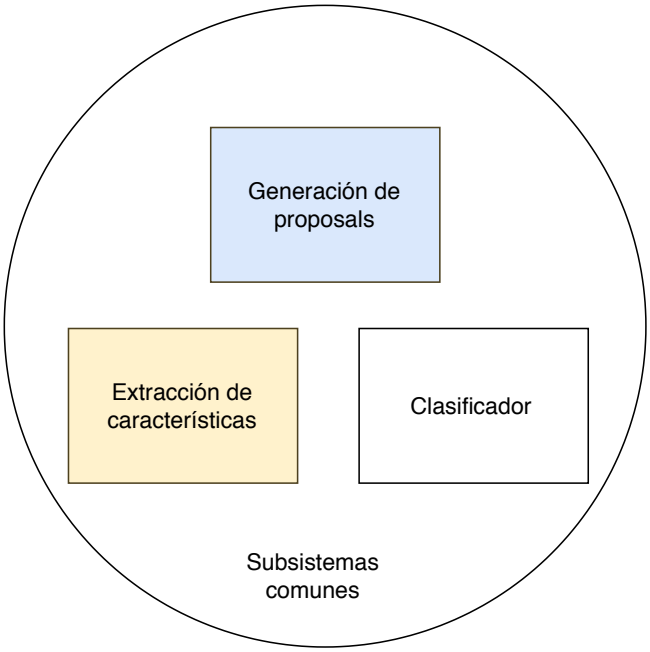

Figura 2.7: Subsistemas comunes.

Figura 2.8: Ejemplo de Selective Search[1].

El funcionamiento del *Selective Search* es sencillo [1]. En primer lugar, se realiza una segmentación profunda de la imagen. La sobre-segmentación de la imagen nos implica que la mayoría de las partes contengan información repetida, por lo que, en procesos iterativos, los segmentos con las mismas características se van uniendo. Los segmentos son estudiados desde cuatro tipos de semejanza en este algoritmo, y son las similitudes de color, textura, tamaño y forma. En la similitud del color se obtienen los

histogramas de cada segmento y se interseptan. En la similitud de la textura, se extraen los derivados Gaussianos para 8 orientaciones y se interseptan el histograma para cada canal RGB y orientación de los segmentos. En la similitud de tamaño, las regiones pequeñas se unen con mayor rapidez por lo que se reduce el tiempo de ejecución. En la similitud de formas, se mide el parecido de los segmentos intentando que ambos coincidan espacialmente. Finalmente, se genera una similitud final que es la suma de cada similitud ponderada. Este parámetro es el usado en la unión.

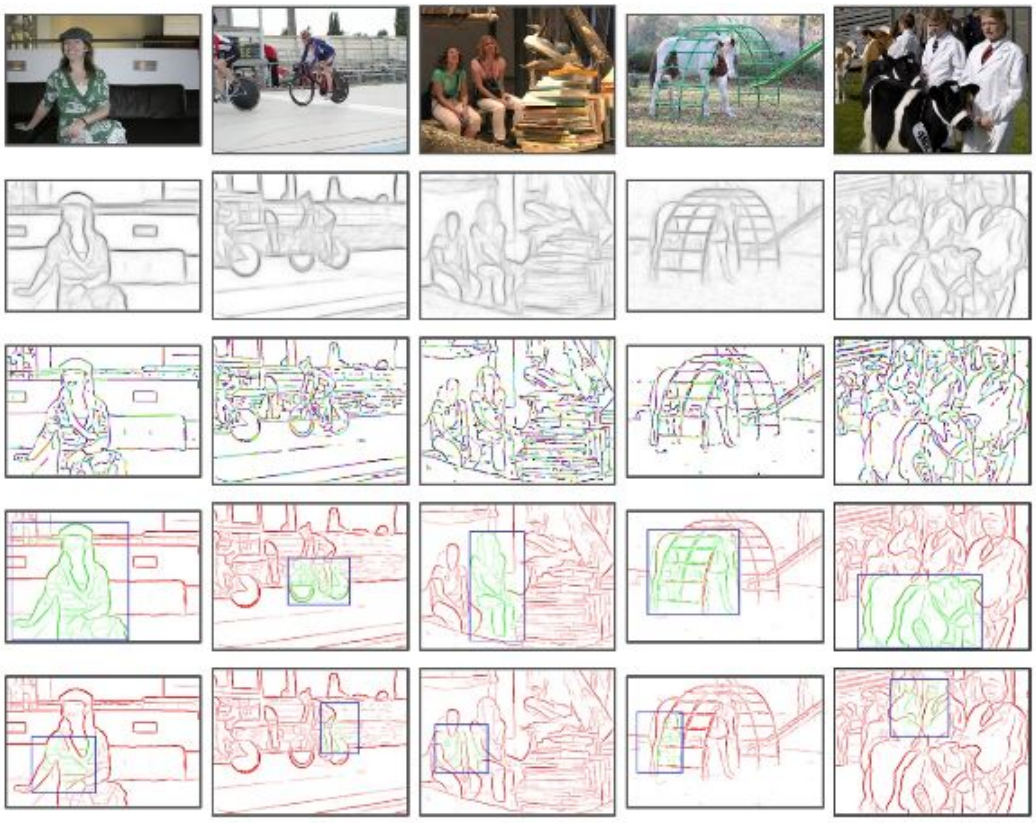

Figura 2.9: Ejemplos de Edge Boxes[2].

Otro algoritmo que es de especial interés es el *Edge Boxes* [2]. Este algoritmo tiene como fundamento que la cantidad de contornos que encierre una región es proporcional a la probabilidad de que exista un objeto dentro de la misma ROI. Para poder puntuar cada región se mide el número de aristas que contiene la ROI, exceptuando aquellas que se encuentren en los límites de la región. El funcionamiento general sigue las siguientes etapas como se muestra en la Figura 2.9:

1. Partiendo de la imagen, se calcula la estructura de bordes (mapa de bordes).

2. Se generan grupos de bordes en la estructura.

3. Se generan ROIs a partir de los grupos, y se generan los *proposals* a partir de las regiones con mayor puntuación.

Además, como *Edge Boxes* implementa estructura de datos eficientes, se puede analizar una gran cantidad de regiones por segundo.

2.6. Seguimiento de fuentes transmisoras

Una vez detectada la fuente transmisora, se pasa a la etapa de seguimiento, en la que se persigue al objeto deseado a lo largo del tiempo utilizando la información de uno o varios *frames* anteriores.

Los algoritmos de seguimientos aplican dos modelos distintos en su funcionamiento, estos son los modelos de movimiento y de apariencia.

En el modelo de movimiento, según la velocidad y la dirección del desplazamiento del objeto en *frames* anteriores, se realiza una estimación gruesa de donde es probable que se encuentre el objeto en el *frame* actual.

En el modelo de apariencia, estudiando las características del objeto (como color, bordes u otro), se realiza el ajuste fino respecto al ajuste grueso del modelo de movimiento.

Algunos algoritmos de seguimiento son:

- *BOOSTING* [23], basado en el clasificador supervisado AdaBoost. En el que se va entrenando según el paso del tiempo con la adquisicón de nuevas ROIs.

- *MIL* [24], similar al algoritmo BOOSTING pero en vez de alimentar al clasificador con las ROI, se alimenta con un vecindario colindante al objeto.

- *KFC* [25], a partir de los vecindarios descritos en el MIL, se estudia las zonas superpuestas.

- *TLD* [26], se divide en tres etapas: seguimiento, encargado de seguir; detección, se estudia el objeto y se corrige el seguimiento; y aprendizaje, estima los errores del detector y lo actualiza.

- *Median Flow* [3], estudia la coherencia temporal de la trayectoria (supuesta), esto es, se estudia cómo avanza la trayectoria de un punto hacia adelante y hacia atrás

en el tiempo. Por esto, el algoritmo *Median Flow* en primera instancia rastrea un punto hacia adelante en el tiempo (*frames*). Seguidamente, con la posición final del punto, se obtiene la trayectoria en hacia atrás en el tiempo (*frames*). Finalmente, se obtiene la diferencia entre trayectorias y si difieren significativamente, se descarta la trayectoria hacia adelante ya que se considera errónea. En la Figura 2.10 se observa un ejemplo, donde las trayectorias hacia adelante y hacia atrás del punto 1 coinciden, pero no las trayectorias del punto 2 debido a la oclusión. El error entre ambas trayectorias descarta esa trayectoria. Cabe destacar que este algoritmo presenta un alto rendimiento para objetos planos.

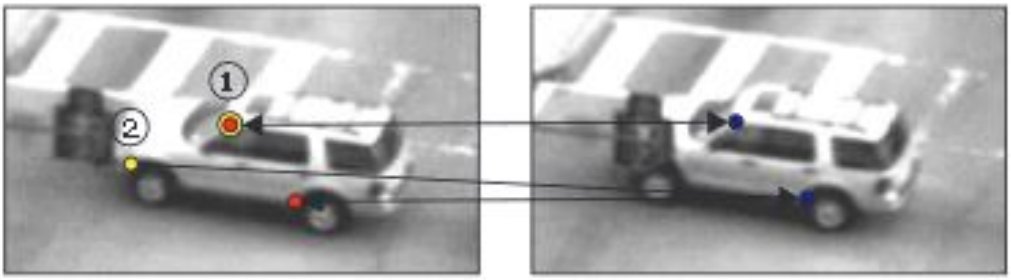

Figura 2.10: Ejemplo de Median Flow[3].

Capítulo 3

Diseño de la solución propuesta

En este capítulo se expone el diseño de la solución propuesta, donde se plantea un sistema que intenta resolver las dificultades que presentan los sistemas OCC cuando son incorporados a dispositivos *wearables* o portables.

Esta solución utiliza detección espacial, aprovechando la diversidad espacial de las cámaras. La detección de la fuente se realiza aprovechando las características relacionadas con su forma.

El presente capítulo se divide en dos secciones. En primer lugar, se expone la arquitectura del transmisor, y seguidamente se discute la arquitectura del receptor. Sin embargo, en las decisiones tomadas en el diseño de los dos sistemas se consideran varios aspectos relacionados con la naturaleza del canal y las técnicas que debe realizar el receptor para la correcta adquisición de la señal: desde la técnica de detección utilizada (detección espacial), hasta el esquema de modulación y codificación de la señal.

Naturaleza del canal y la fuente

La naturaleza de la fuente como objeto emisor luminoso es determinante en el diseño del sistema, ya que nos ofrece la oportunidad de discriminar objetos en la imagen que no emiten luz. Si se explota este fenómeno, se puede realizar una extracción efectiva de esta característica de la fuente. Esto significa que es posible modificar la ganancia y el tiempo de exposición en la recepción para que las señales luminosas debidas a las reflexiones de luz en los objetos de la escena se exprese mínimamente en la imagen, es decir, no se aprecien en la imagen. Esto resulta beneficioso puesto que el receptor encontrará más fácilmente la fuente transmisora, en una imagen oscura. En la Figura 3.1

se muestra una imagen con un tiempo de exposición bajo y una ganancia relativamente baja.

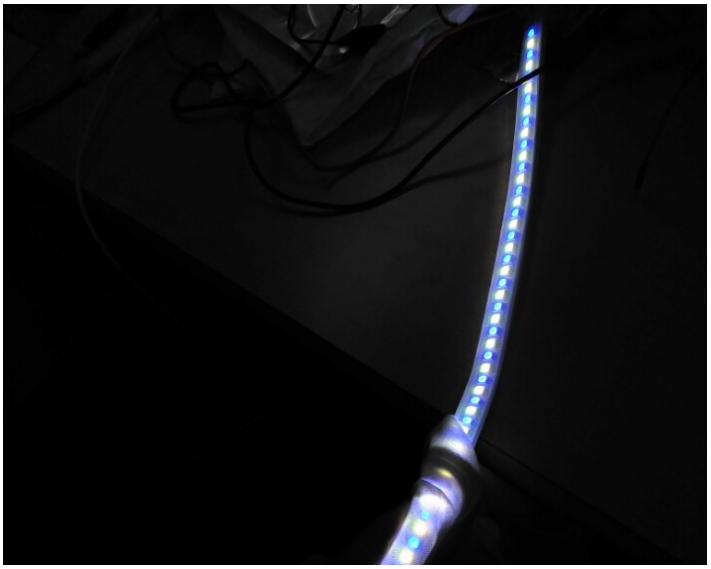

Figura 3.1: Tiempo de exposición bajo.

La imagen oscura implica que existen menos objetos reconocibles, con lo cual, menos interferencias en la detección. Esto se traduce en una mejora en la Relación Señal a Interferencia.

Cabe destacar que existe una relación de compromiso entre la ganancia y el tiempo de exposición de la cámara. De reducir el tiempo de exposición, los efectos de *Rolling Shutter* discutidos en el capítulo 2 son más pronunciados. Esto puede llevar a que se produzcan mezclas parciales entre símbolos dentro de un mismo *frame*. Por tanto, es mejor en la medida de lo posible, modificar únicamente a la ganancia.

Adaptación de la frecuencia de transmisión a los fps

En el diseño de este sistema se considera que durante la transmisión de una trama de datos (modulados espacialmente), al menos uno de *frames* debe contener la trama completamente.

Dado el modo de adquisición común en las cámaras comerciales (*Rolling Shutter*), la frecuencia de muestreo de la señal es mucho mayor a la tasa de captura de imágenes de la cámara, debido a que la imagen se escanea línea a línea de píxeles[1]. Sin embargo, en este caso, ya que se desea que la imagen contenga una trama de datos completa

[1]la frecuencia de muestreo está relacionada con el intervalo entre el escaneo de filas y su tiempo de escaneo

para un frame, se debe mantener la transmisión de la trama espacial durante dos o más tiempos de *frame*.

Por tanto, es necesario seleccionar la frecuencia de transmisión de la fuente. En el caso de utilizar una modulación OOK, donde se envía intensidad luminosa para el dato binario 1, o ausencia para el dato binario 0, la frecuencia de transmisión coincide con la frecuencia de conmutación de los LEDs del transmisor.

De mantener la trama durante un tiempo de captura de *frame*, sería necesario utilizar mecanismos de sincronización dentro del *frame* para obtener los datos.

En este caso si el receptor no está perfectamente sincronizado, el *frame* obtenido no corresponde a una única trama, sino a la interferencia entre la trama actual y la anterior o posterior. Esta interferencia se puede observar en la Figura 3.2.

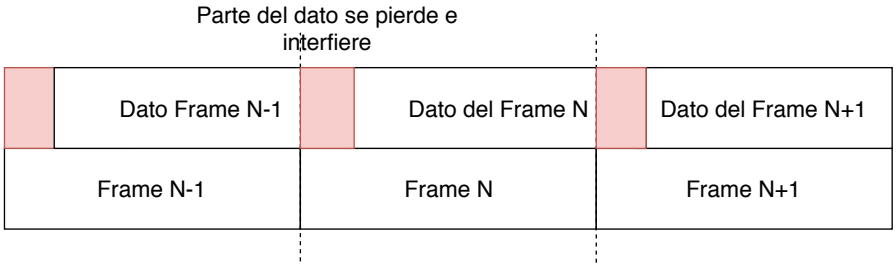

Figura 3.2: Ejemplo de desincronización e interferencia.

De mantener la trama durante dos o más veces el tiempo de *frame*, siempre existirá un *frame* con la trama sin interferencia tal y como se observa en la Figura 3.4. Con lo

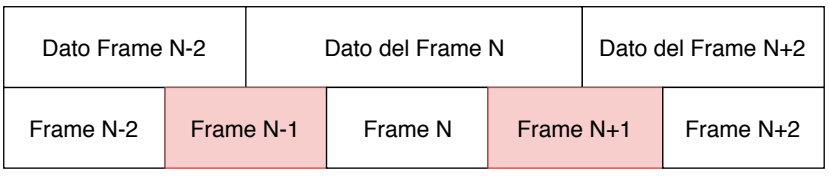

Figura 3.3: Ejemplo de comunicación con una tasa de Nyquist de 2.

cual nuestra frecuencia de transmisión tiene que cumplir la Ecuación 3.1.

$$F_{tx} \leq \frac{F_{fps}}{2} \tag{3.1}$$

Dado que la frecuencia de transmisión está relacionada con la tasa de captura de *frame*, en términos de nomenclatura, se utiliza la tasa de Nyquist t_N que está referido al tiempo de *frame* $(1/F_{fps})$. Por tanto, la tasa de Nyquist sigue la expresión de la Ecuación .

$$T_{tx} = \frac{t_N}{F_{fps}} \tag{3.2}$$

donde $t_N \geq 2$.

En el ejemplo de la Figura 3.4, para una tasa de Nyquist de 2, el dato se mantiene el doble del tiempo de *frame*. En este caso, se puede observar que los *frames* N-1 y N+1 obtienen un dato inválido que se corresponde con la interferencia entre tramas consecutivas. Por este motivo, en la decodificación y demodulación, se debe determinar cuáles son los *frames* con información válida.

Por tanto, es necesario usar técnicas de sincronismo para la detección correcta de los *frames* válidos.

Además, cabe señalar que cuanto mayor sea la tasa de Nyquist, mayor será el número de *frames* válidos para cada trama de datos transmitida (información redundante) pero el número de *frames* no válidos permanecerá constante. Por esto, con el incremento de la tasa de Nyquist, se reduce la velocidad de transmisión.

Esquema de la solución

El esquema general del sistema se muestra en la Figura 3.4. El sistema transmisor es el responsable de generar los datos, codificarlos, modularlos y transmitirlos. Se divide en: sistema de control, codificador, modulador, driver LED y transmisor LED.

Específicamente el sistema de control se encarga de generar los datos que utiliza el codificador y los posteriores subsistemas y coordinar las interfaces de entrada/salida. Además, los datos se pueden obtener a través de un sistema de generación de datos aleatorios o de las mismas interfaces de entrada/salida.

El sistema receptor es el responsable de obtener las imágenes, pre-procesarlas, adquirir la señal y presentar la información. Por esto se divide en los siguientes subsistemas: adquisición del frame, preprocesamiento, adquisición de la señal y presentación. En el preprocesamiento se realizan operaciones básicas sobre la imagen obtenida para prepararla para la demodulación. Un ejemplo sería el filtrado paso alto para obtener

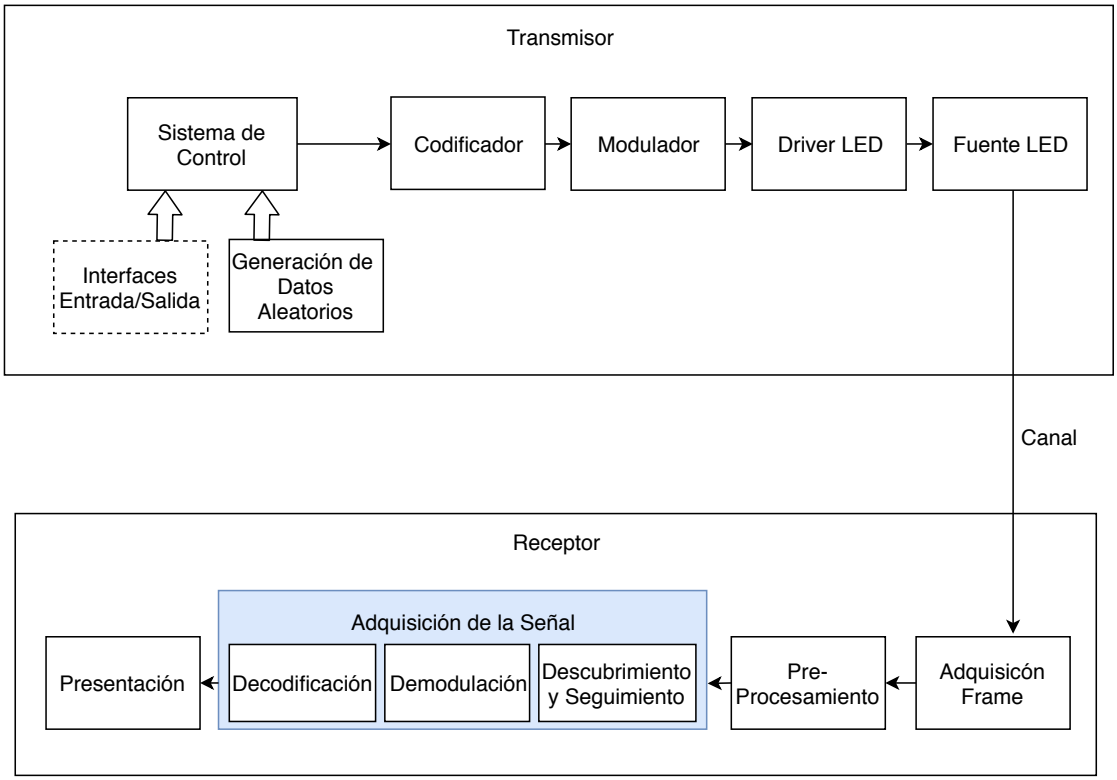

Figura 3.4: Esquema general del sistema de comunicación.

los bordes. En la adquisición de la señal se realizan las operaciones de descubrimiento y seguimiento de la fuente transmisora, además de la demodulación y decodificación de la señal.

El estudio del sistema receptor en este trabajo se centra exclusivamente en las etapas de descubrimiento y seguimiento, ya que supone una novedad en las comunicaciones OCC para los sistemas móviles. Por este motivo, no se diseñan las etapas de decodificación y demodulación, aunque se tienen en consideración a la hora de diseñar el transmisor en su totalidad.

3.1. Transmisor

En esta sección se estudian las partes que componen el transmisor. Inicialmente, se discute el diseño de la fuente LED y le sigue la presentación del diseño de los demás subsistemas en el sentido contrario al flujo de la información.

3.1.1. Fuente LED

La fuente LED es el sistema encargado de convertir las señales eléctricas en señales ópticas. Consiste en una matriz de NxMLEDs. Cada LED tendrá una función asociada a la transmisión o a la ayuda en la localización de la fuente.

3.1.2. Distribución de la Matriz LED

Cada LED se compone de tres canales RGB (rojo, verde y azul), por tanto, el número de canales disponibles es de NxMx3, siendo N el número de filas de LED y M el número de columnas.

Dada la relación que existe entre los LEDs y el filtro de Bayer de la cámara, es posible utilizar estos canales de forma independiente, que pueden ser utilizados tanto para la transmisión como para cualquier otra función.

3.1.2.1. Uso del canal verde

De los tres canales disponibles, se sacrifica uno para enviar un valor constante. Esto facilita la labor del receptor en la detección del transmisor, ya que, si se analiza el canal con valores constantes, se obtiene una imagen donde los valores del transmisor a descubrir no varían o varían muy poco con el tiempo. Por tanto, de la matriz inicial se tiene que la matriz útil de transmisión es de tamaño NxMx2.

Para poder elegir el canal que se mantendrá constante, se debe analizar la respuesta espectral del filtro de Bayer de la cámara. La transmitividad del filtro de Bayer no es perfecta y tiene un ancho de banda para cada canal, por lo que parte del espectro perteneciente al fotorreceptor verde lo capta el azul y otra parte el rojo. Esto es, existe una interferencia espectral en la recepción. Además, el fotorreceptor verde también capta longitudes ondas pertenecientes al canal azul y al rojo. En la Figura 3.5 se presenta la transmitividad espectral de la cámara PiCameraV2.

Se puede observar que la interferencia más significativa se da cuando se transmite en el verde, ya que los canales rojo y azul le interfieren. Por esto, se elige el canal verde para la transmisión del valor constante. Al transmitir un valor constante, no nulo, la interferencia al resto de canales no varía con el tiempo, con lo cual es posible eliminarla en el proceso de demodulación.

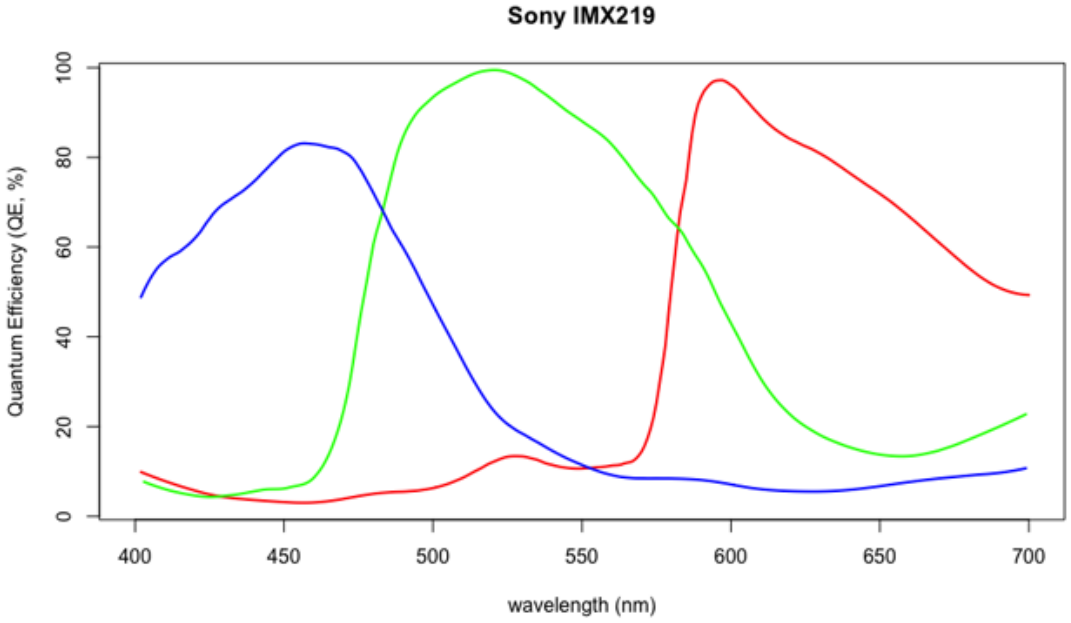

Figura 3.5: Transmitividad espectral de la cámara PiCameraV2 [4].

En definitiva, se usa este valor constante, para evitar que la transmisión de datos afecte a la localización de la fuente en el receptor. En este sentido, se consigue que las tramas con símbolos en ausencia de luz, no implique la pérdida de la forma de la matriz.

3.1.2.2. Anchor LED

Por otra parte, en la decodificación y demodulación, es importante conocer la rotación del transmisor, para poder observar y clasificar los datos correctamente. Por esto, se usan los LEDs de ancla o anchor LED.

Los LED de ancla se distribuyen espacialmente en la matriz siguiendo una configuración específica.

Para poder conocer la rotación se busca una figura no simétrica que posea un grado de libertad. Se llama grado de libertad a los grados de rotación, respecto al centro, que tiene que sufrir la figura para que coincida con la original. Por ejemplo, si un cuadrado se rota 90, 180, 270 y 360° respecto a su centro, se obtiene el mismo cuadrado. En caso de escoger una figura con un solo grado de libertad, se sabe cuál es la orientación.

Con un solo anchor LED se tiene un punto y no se puede saber la rotación, ya que existen infinitos grados de libertad. Con dos anchor LEDs se tiene una recta con dos grados de libertad. Los dos ángulos posibles de rotación dependen de la pendiente

de la recta. Con estos dos grados no sabemos con exactitud cuál es la orientación de la figura. Con tres anchor LED se tiene un triángulo. En el caso de dibujar un triángulo equilátero, se obtienen tres grados de libertad. Sin embargo, con un triángulo no equilátero, se tiene solo un ángulo de libertad y, por tanto, se sabe el ángulo de rotación. Esto se debe a que existe al menos un lado que no es igual al resto, se pierde la simetría.

En el diseño del transmisor, ya que se tiene una matriz NxMx3. Para cumplir esta condición de triángulo no equilátero, se eligen tres LEDs en tres esquinas distintas de la matriz. Esto produce un triángulo rectángulo escaleno, ya que uno de los ángulos es recto. Cabe señalar que esta no es la única solución, pero se toma por su simpleza.

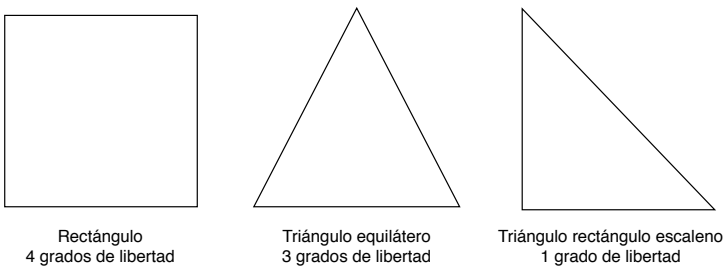

Figura 3.6: Ejemplos de figuras geométricas y sus grados de libertad.

Además, para poder diferenciar los anchor LEDs frente a los LEDs de datos, los anchor LED emiten con mayor brillo o intensidad luminosa. De esta forma, variando el umbral de la binarización de la imagen, se puede determinar la posición de los tres anchor LEDs.

3.1.2.3. LEDs de entrenamiento

Otros factores importantes a tener en cuenta en la elaboración de un transmisor OCC son los LEDs de entrenamiento. Estos LEDs son usados para poder decodificar y demodular de forma correcta. Gracias a estos tipos de LEDs, podemos conocer las interferencias que sufren los canales RGB entre sí y estimar la respuesta del canal. Las interferencias a analizar son la verde-azul, la verde-rojo y la azul-rojo analizada con una transmisión en todos los canales (luz blanca). Además, también se transmite un LED

con el canal verde sin presencia del resto. Este LED nos permite modelar y comparar la influencia de este canal frente a los demás. Por tanto, los LEDs de entrenamiento que se usan son:

- LED blanco (todos los canales encendidos): interferencias Rojo - Verde - Azul.
- LED amarillo: interferencias Rojo - Verde.
- LED cian: interferencias Verde - Azul.
- LED verde.

3.1.2.4. Distribución Planteada

Según lo indicado, se necesitan 4 LEDs de entramiento y 3 anchor LEDs. En el diseño propuesto, para obtener un mayor número de LEDs de datos, se usan los tres anchor LEDs como LEDs de entrenamiento. Es requisito que el canal verde sea constante y de mismo valor para todos los LEDs, lo que se cumple siempre en los LEDs de entrenamiento planteados. Se elige el Blanco, Rojo-Verde y Verde-Azul para los anchor LEDs, y el LED con el canal verde se posiciona en la esquina sobrante.

Por tanto, la distribución diseñada tiene la forma que se muestra en la Figura 3.7.

Cabe señalar que la posición de los LEDs de ancla y de entrenamiento se pueden modificar, siempre y cuando se cumpla que la figura descrita por los anchor LEDs solo tenga un grado de libertad, y existan como mínimo 4 LEDs de entrenamiento.

En el diseño implementado se tiene la matriz inicial con un total de $N*M*3$ LEDs, presenta un número de LEDs de datos igual a la expresión de la Ecuación 3.3.

$$N * M * 2 - N_{anchor} * 2 - N_{entrenamiento} = N * M * 2 - 2 * 3 - 2 * 1 \qquad (3.3)$$

Donde N_{anchor} es el número de leds de ancla, y $N_{entrenamiento}$ el número de leds de entrenamiento.

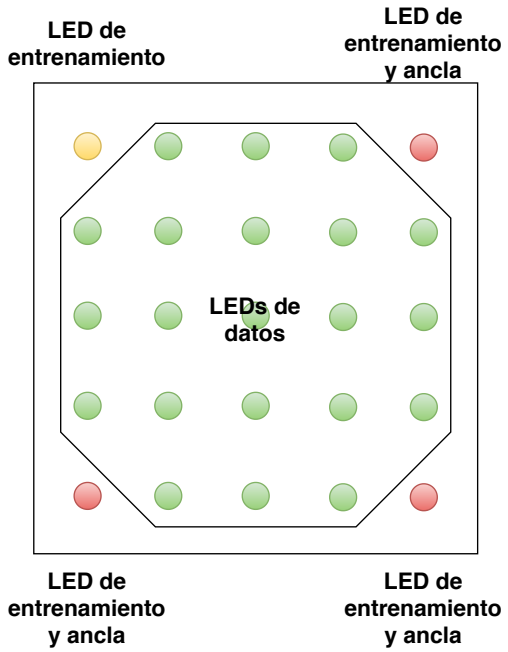

Figura 3.7: Distribución de la matriz LED.

3.1.2.5. Velocidad de Transmisión

La tasa binaria en relación a la tasa de Nyquist, los FPS y la dimensión de la matriz viene determinada por la Ecuación 3.4.

$$B_t = (2 * N * M - 8)(FPS/t_N) \tag{3.4}$$

Donde:

- B_t es la velocidad de transmisión en bits por segundo.
- N son las filas de la matriz de LEDs.
- M son las columnas de la matriz de LEDs.
- t_N es la tasa de Nyquist.
- FPS son los Fotogramas Por Segundo del sistema.

Cabe destacar que cuanto mayor sea la dimensión de la matriz, mayor será la tasa binaria; lo mismo ocurre cuanto menor sea la tasa de Nyquist y cuanto mayor sean los FPS del sistema.

Una vez comprendida la distribución de los LEDs, se diseñan el resto de subsistemas.

3.1.3. Driver LED

El driver LED es el sistema encargado de obtener las señales eléctricas necesarias para alimentar a los LEDs. Como su nombre indica, conduce a los LEDs con la señal eléctrica modulada en corriente.

Otra función del driver LED es mantener al LED en la zona lineal o la zona de corte. Si estuviera en zona de saturación, la corriente inducida no aumentaría al aumentar la tensión y, por tanto, no produciría más fotones al incrementar la tensión. Esto se traduce en una eficiencia menor.

3.1.4. Modulador

Seguidamente, se encuentra el modulador. Este dispositivo es el responsable de generar la señal eléctrica modulada a partir de las tramas de datos. Se usa la modulación OOK espacialmente distribuida. este subsistema es el responsable de generar la tensión adecuada para valores lógicos uno y cero binarios. Normalmente se genera una tensión nula para el dato binario 0, y tensión no nula para el dato binario 1.

3.1.5. Codificador

El codificador es el sistema responsable de generar las tramas a partir de los datos. Por tanto, este sistema es el que adapta los datos a la configuración de la matriz LED. Por esto, debe conformar las tramas de acuerdo con los siguientes puntos:

- Se añade la restricción de que el canal verde siempre se encuentre encendido.
- Se posicionan los LED de entrenamiento y anchor LED en los extremos de la matriz.
- Se especifica una mayor intensidad en los LEDs de ancla y una menor intensidad en los LEDs de datos.
- Se especifican los 1 y 0 a partir de los datos para que el modulador genere las señales eléctricas.

Además, las tramas se generan en intervalos de la tasa de Nyquist a partir de los FPS del sistema. Recordar que esto se realiza para poder obtener un dato válido en el receptor sin que sufra interferencia.

3.1.6. Sistema de control e Interfaces I/O

El sistema de control es el encargado de supervisar las interfaces de entrada y salida y el sistema de generación de datos aleatorios, así como entregar los datos al codificador.

3.1.7. Generación de datos aleatorios

Este sistema se encarga de generar datos aleatorios en vez de obtenerlos de las interfaces de entrada a través del sistema de control (sistema útil para la evaluación del sistema de comunicación).

3.2. Receptor

El receptor es el sistema encargado de recibir y presentar la información obtenida, y se compone de los siguientes subsistemas: adquisición del frame, preprocesamiento, adquisición de la señal y presentación.

En la adquisición del *frame*, se obtiene la imagen a partir de la cámara. En el preprocesamiento, se aplica algún tipo de preprocesado al *frame*, como puede ser cambiar el contraste o un filtrado paso alto para obtener los bordes. En la adquisición de la señal, se descubre y sigue, demodula y decodifica la fuente transmisora. Finalmente, en la etapa de presentación, se muestran los datos obtenidos.

Ya que el estudio principal del receptor son las etapas de descubrimiento y seguimiento dentro de la adquisición de la señal, no se diseñan ni el decodificador, ni el demodulador. Por tanto, el diseño propuesto consta de las siguientes etapas: adquisición del frame, descubrimiento y seguimiento, y presentación.

En este caso no es necesario realizar el preprocesamiento ya que no se precisa de ningún cambio en la imagen, solo bajar el tiempo de exposición que se realiza directamente en la adquisición del frame. Finalmente, destacar que en la etapa de presentación solo se muestran los datos obtenidos en el descubrimiento y seguimiento.

La Figura 3.11 muestra los distintos subsistemas diseñados en el receptor.

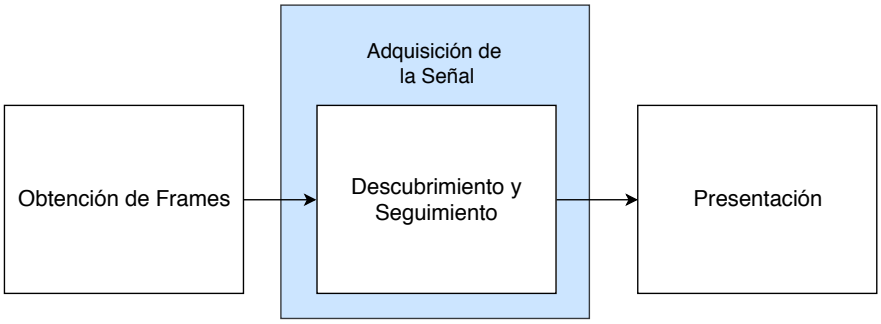

Figura 3.8: Esquema del receptor diseñado.

3.2.1. Adquisición del frame

La adquisición del frame es el subsistema encargado de adquirir el *frame* que entrega la cámara. Este *frame* es pasado a la etapa de adquisición de la señal para su análisis.

En la introducción del capítulo se habla sobre la naturaleza del canal y fuente, donde el objeto buscado, el transmisor, es una fuente de luz por sí misma, y el entorno es poco ruidoso.

Para ello se baja el tiempo de exposición de la cámara y se disminuye la sensibilidad de la cámara, con lo que se consigue oscurecer todo aquello que no transmite luz por sí mismo. Tras un proceso de binarizado de la imagen, se destacan fácilmente las fuentes luminosas. Con lo cual, la cámara elegida debe permitir variar el tiempo de exposición y su ganancia.

3.2.2. Descubrimiento y Seguimiento

En la actualidad, descubrir la fuente transmisora en cada frame es computacionalmente costoso, y, por tanto, también lo es en tiempo. Además, los algoritmos actuales también presentan una sensibilidad baja de detección como es el caso del YOLO.

Para solucionar los problemas presentados, se hace uso de dos etapas bien diferenciadas: el descubrimiento y el seguimiento.

El descubrimiento, como su nombre indica, es el encargado de localizar la fuente. Esta etapa busca la fuente transmisora en todos los *frames*. Una vez descubierto el *frame* donde se encuentra el transmisor, se pasa la ROI que contiene a la fuente al seguimiento. La ROI del descubrimiento se denomina ROI descubrimiento.

En el seguimiento, se obtiene la ROI descubrimiento y se inicializa esta etapa. La ROI descubrimiento que se selecciona se denomina ROI seleccionada. Para cada nuevo

frame en el seguimiento se devuelve una ROI donde es muy probable que se encuentre el objeto seguido, la fuente transmisora en este caso. Para ello, se analiza la información de las ROI de uno o varios *frames* anteriores. La ROI devuelta por el seguimiento se denomina ROI seguimiento. Por tanto, tenemos tres tipos de ROIs:

- **ROI descubrimiento**: la ROI devuelta por el descubrimiento.
- **ROI seleccionada**: la ROI se utiliza para iniciar el seguimiento.
- **ROI seguimiento**: la ROI devuelta por el seguimiento a partir de la información recogida en uno o varios *frames* anteriores.

Además, dado que la ROI seguimiento se va deteriorando con el tiempo, cada cierto periodo se debe descubrir nuevamente la fuente.

Las interacciones entre estas dos etapas se muestran en grafo de estados en la Figura 3.11. Cada interacción se explica en los puntos a continuación.

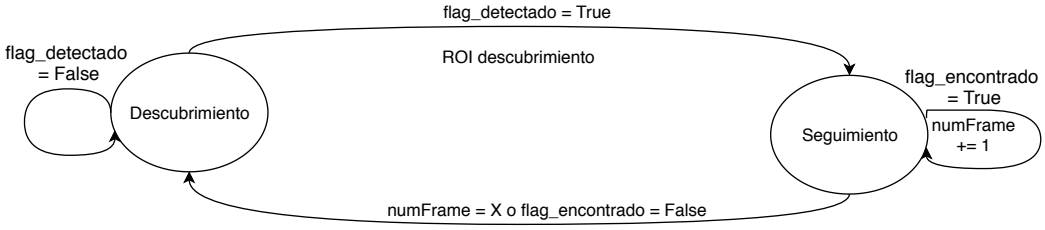

Figura 3.9: Diagrama de flujo de la adquisición de la señal.

3.2.2.1. Descubrimiento

Dado un frame de un vídeo, el descubrimiento es la etapa que genera una ROI donde se encuentra la fuente transmisora. Para ello, se utiliza la detección espacial de la fuente. Es decir, se usan las propiedades físicas del objeto a descubrir para su detección.

La solución propuesta para el descubrimiento, más centrada en las comunicaciones OCC, y que difere en las soluciones anteriores, es lo que se presenta en la Figura 3.11.

Como se ha indicado, a la hora de la obtención del *frame*, la cámara es configurada con un tiempo de exposición bajo. Intrínsecamente se trata de una extracción de característica del transmisor, y facilita la detección de la fuente. De la imagen obtenida se procesa únicamente el canal verde, canal que mantiene constante la iluminación de

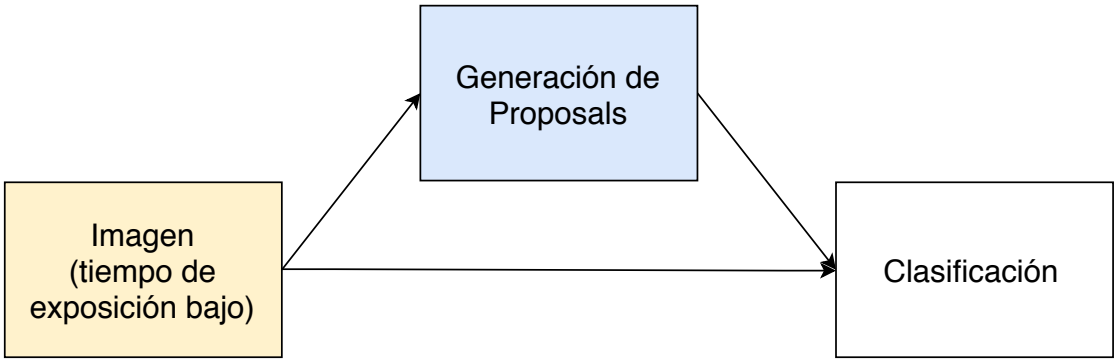

Figura 3.10: Esquema del diseño del descubrimiento.

todos los LEDs. Sobre esta imagen se identifican las regiones propuestas o *proposals*, ROIs donde es posible encontrar la fuente a localizar. Esta etapa es el punto crucial, ya que se pasa de analizar imágenes enteras, a analizar únicamente fragmentos.

Finalmente, de la lista de proposals y la propia imagen, se extraen las características principalmente relacionadas con la forma del emisor (ya que el diseño de la fuente LED no sigue una estructura común). Finalmente, se clasifican las características para cada ROI y se determina si algún *proposal* contiene al transmisor. En caso de que no se descubra la matriz de NxM LEDs, se obtiene una nueva imagen y se realiza el mismo procedimiento. Este proceso se repite en bucle hasta que se descubre un *proposals* que contiene al transmisor. En este momento, el *proposal* se convierte en la ROI descubrimiento, se adapta para la fase de seguimiento y se inicia este.

En el descubrimiento, el tiempo de ejecución varía según el algoritmo de generación de *proposals* y el método seleccionado para la clasificación. Para los algoritmos de generación se consideran los dos propuestos en el capítulo 2. El *Selective Search* puede llegar a suponer un tiempo de computación de 10 segundos, mientras que el *Edge Boxes*, para la misma imagen, supone un tiempo de 0.2 segundos. Además, ambos algoritmos poseen la misma sensibilidad (87 %). Se entiende por sensiblidad la capacidad de no fallar en la detección cuando la fuente está presente en la imagen. Por las evidencias a priori de su funcionamiento, se elige el algoritmo de generación de proposals *Edge Boxes*. Por otro lado, para la clasificación se diseña un algoritmo que detecta y enumera los contornos dentro de la región de evaluación utilizando binarización y detección de bordes.

3.2.2.2. Seguimiento

Una vez se obtiene la ROI descubrimiento, se inicializa el algoritmo de seguimiento. En caso de no poder inicializar el algoritmo, se vuelve a la etapa de descubrimiento.

Como se indica en puntos anteriores, el seguimiento se ejecuta con mayor rapidez. Esto se debe a que los algoritmos evalúan una región donde es probable que se encuentre la fuente seguida en base a la información obtenida sobre ella en *frames* anteriores. Por tanto, este algoritmo devuelve una ROI donde es muy probable que esté contenida.

Cabe señalar que inicialmente la ROI obtenida del seguimiento se encuentra centrada y perfectamente ajustada respecto a la fuente. Sin embargo, con el paso del tiempo, esta centralidad se deteriora, lo que puede producir que el seguimiento la pierda. Además, el área de ROI la seguimiento se ve afectada por el movimiento de la fuente, y tiende a contraerse sobre la fuente o expandirse. Por esto, cada cierto tiempo es necesario volver a descubrir. La evolución de la centralidad y la escala determinan el tiempo de vuelta al descubrimiento, y se analiza en el capítulo 5.

3.2.2.3. Ajuste de la ROI seleccionada

En la etapa del seguimiento se comenta la centralidad de la ROI seguimiento frente a la ROI descubrimiento. Por lo que se supone que la etapa del descubrimiento devuelve una ROI totalmente centrada. Esta suposición no es correcta, ya que los algoritmos de generación de *proposals* devuelven, en la mayoría de los casos, una ROI no centrada.

Por tanto, aunque en teoría el sistema expuesto en los apartados anteriores sea viable, en la práctica dicho sistema presenta dificultades. Con lo cual, es necesario una etapa intermedia que ajuste la ROI descubrimiento a la matriz de LEDs.

Por otra parte, esta etapa aplica un margen de protección, delta, a la ROI ajustada. Este margen da libertad al seguimiento para que pueda seguir al objeto deseado con más precisión. La ROI resultante se denomina ROI seleccionada. El estudio de este margen de protección también se discute en el capítulo 5.

El esquema general resultante es el que se expone en la Figura 3.11.

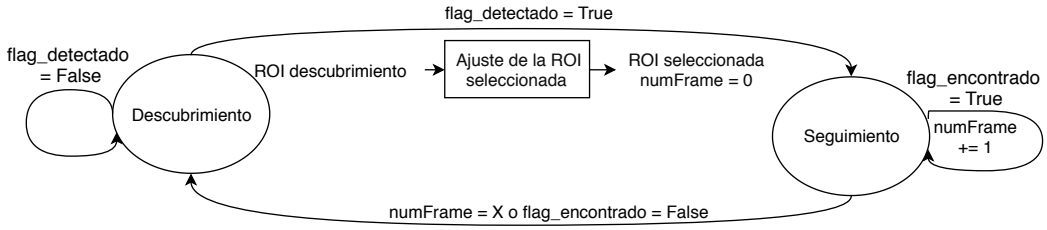

Figura 3.11: Diagrama de flujo modificado de la adquisición de la señal.

3.2.3. Presentación

Finalmente, en el receptor diseñado, se encuentra el subsistema de presentación. A diferencia del sistema de presentación en el esquema general que presenta los datos obtenidos de la decodificación; en este diseño, la presentación dibuja la ROI resultante del seguimiento en la imagen en el caso de que exista, y en caso contrario, solo presenta la imagen.

Capítulo 4

Implementación

En este capítulo se expone la implementación del diseño propuesto en el capítulo anterior. La estructura de este capítulo es similar a la estructura del capítulo de diseño. En primer lugar, se presenta la implementación del transmisor. Posteriormente, se presenta la implementación del receptor.

Cabe recordar que la implementación del transmisor debe estar enfocada hacia los dispositivos *wearables*. Por lo que el transmisor debe ser ligero y de bajo coste y consumo.

4.1. Transmisor

En este apartado se recoge la implementación del transmisor. El transmisor consta de las siguientes partes: la fuente LED, el driver LED, el modulador, el codificador, y la interfaz para la generación y recepción de datos donde se encuentre el sistema de control.

En este caso, el esquema del transmisor implementado se muestra en la Figura 4.1.

Se puede observar que el driver LED y el transmisor LED están contenidos en el bloque AddresableRGB, mientras que los demás subsistemas se contienen en el NUCLEO-L432KC.

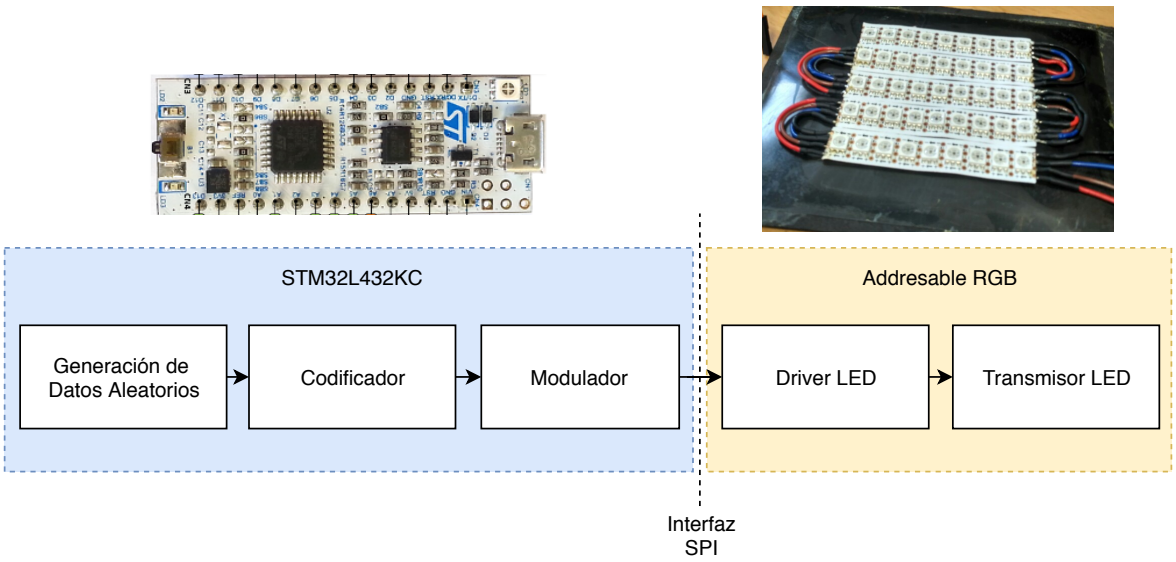

Figura 4.1: Esquema general de la implementación del transmisor.

4.1.1. Addresable RGB

El AddresableRGB realiza las funciones de la fuente LED y el driver LED. En esta implementación, la fuente LED es una matriz de dimensiones 5x5 LEDs, con un total de 25 LEDs RGB. Por otra parte, el driver LED es el responsable de adaptar las señales eléctricas de la entrada a las tensiones necesarias para el funcionamiento de cada LED.

Se elige la tira de LEDs RGB direccionables NeoPixel que cuentan basado en el controlador integrado APA102.

Esta tira LED contiene una serie de LEDs RGB APA102 (tipo SMD5050RGB) con un driver integrado para su control de LED. Cada APA102 de la tira ofrece una interfaz de entrada y salida con 6 pines digitales:

- V+, tensión de alimentación de 5 Voltios común a todos los LEDs.
- CLK IN, señal de reloj de entrada.
- CLK OUT, señal de reloj de salida.
- DATA IN, señal de datos de entrada.
- DATA OUT, señal de datos de salida.
- GND, referencia a tierra común a todos los LEDs.

Esta ordenación de los pines, permite que los LEDs puedan ser conectados en configuración *Daisy Chain*, esto es, las salidas de datos y reloj de un APA102 ataca las entradas del siguiente, lo que permite adaptar el número de LEDs a la aplicación deseada.

Para la configuración del color y la intensidad de cada uno de los LEDs la tira acepta un protocolo sobre SPI que acepta una trama con los campos indicados en la Figura para el direccionamiento de cada uno de los LEDs.

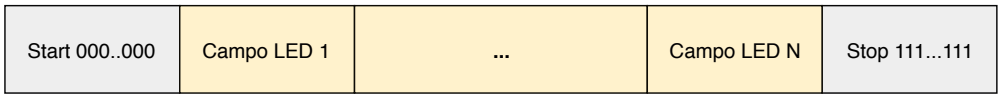

Figura 4.2: Estructura de la trama.

En primer lugar, se envía una cabecera de start de 32 bits a 0, seguido de los campos de configuración para cada LED de 32 bits, y finalmente una cola de stop de 32 bits a 1. La cabecera start indica el inicio y la cola de stop indica el fin. La trama es el conjunto de todos los campos de configuración del LED más la cabecera y la cola. Cabe señalar que como se tiene una matriz de 5x5, la trama debe constar de 25 campos de configuración LED aparte de los delimitadores.

Como se observa, el control de los LEDs es de tipo cascada, ya que no se controlan cada uno de los LEDs por separado, sino que la información circula a lo largo de la tira, y en cada LED se escoge el campo de control necesario en cada caso. De esta forma, cada LED obtiene el campo y deja pasar el resto de la trama sin modificarla. Cuando la trama ha sido correctamente enviada, la cola final transmitida, dura un tiempo tal que indica a los LEDs que adquieran la nueva configuración.

La estructura del campo LED se muestra en la Figura 4.3.

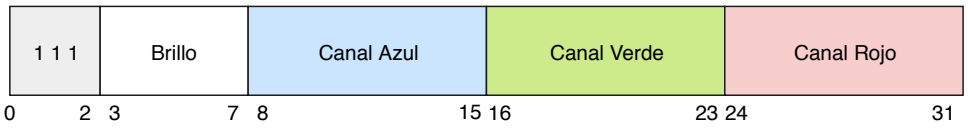

Figura 4.3: Estructura del campo LED.

Los tres primeros bits consisten en un indicador de sincronismo de paquete y siempre deben estar a 1, los siguientes 5 bits indican el brillo general (entendido este como una ganancia que afecta a todos los canales por igual), los siguientes 8 bits indican el valor del canal azul, los siguientes 8 el valor del canal verde, y los siguientes 8 el valor del canal rojo.

Por tanto, estos campos nos definen el color y brillo de cada LED. En el diseño de la matriz, los LED en las esquinas son los LED de entrenamiento y ancla. Estos deben tener un mayor brillo. Así, el codificador y modulador deben generar la trama teniendo esto en cuenta.

Por otro lado, ya que el canal verde siempre está encendido, los campos de 8 bits de color verde deberán estar siempre a 1. Y dado que se usa una modulación OOK, los campos de 8 bits azul y rojo varian entre los valores 255 (todos los bits a 1) y 0, dependiendo del contenido de la transmisión.

4.1.1.1. Generación de la señal de control

Las señales de control de la tira AddressableRGB son: la alimentación, la referencia a tierra, el reloj (CLK) y los datos en serie (DATA).

Para la alimentación se utiliza un generador de tensión. En la práctica la generación de alimentación debe ser una fuente portable tal como baterías o power banks.

Además, dado que la referencia a tierra debe ser común en todos los puntos del sistema, para que se creen bucles de retorno y para que los niveles de la señal estén referenciados correctamente. En definitiva, la referencia a tierra se pone en común con el generador y el dispositivo NUCLEO-L432KC.

Las señales de reloj y datos se generan utilizando una interfaz SPI como se observa en la Figura 4.1. Esta interfaz tiene 4 señales:

- CLK: Señal del Reloj.
- MOSI (Master Out Slave In): Señal que transmite el maestro al esclavo. Se entiende el maestro en este caso al NUCLEO-L432KC y al esclavo a la tira addressable.
- MISO (Master In Slave Out): Señal que transmite el esclavo al maestro, de la tira al NUCLEO-L432KC
- SS: Selección del esclavo.

En este caso, ya que solo existe envío de datos en una dirección, del microcontrolador a la tira, solo se usa la señal MOSI de la interfaz SPI para generar la señal DATA IN de la tira. Por otra parte, se utiliza la señal de reloj CLK necesaria para la tira. Además, no se hace uso de la señal de selección del esclavo puesto que existe un solo esclavo.

4.1.1.2. Fabricación de la matriz

En el diseño planteado se desea una matriz 5x5 LEDs RGB, pero se dispone únicamente de tiras LED. Por tanto, es necesario realizar una adaptación para la fabricación de la matriz.

La tira de LED RGB addressable se puede cortar en cualquier intersección entre APA102. Esto permite realizar una soldadura para cada pin de la intersección como se observa en la Figura 4.4. En caso de cortar cada LED por separado, es necesario realizar 196 soldaduras. Para optimizar la elaboración de la matriz, se divide la tira en 5 tiras independientes de 9 LEDs y se sueldan entre ellas, lo que suponen realizar 28 soldaduras.

Con esta configuración se tiene una matriz de 9x5 LEDs. Para adaptar esta matriz a una configuración de 5x5, lo que se hace es poner a cero el brillo y el color para todos los LEDs intermedios de la tira. Esto es, para cada tira se configura los LED de las posiciones impares (1, 3, 5, 7 y 9). El factor a tener en cuenta en este punto es que la etapa codificación y modulación transmita información en las posiciones relativas a los LEDs intermedios.

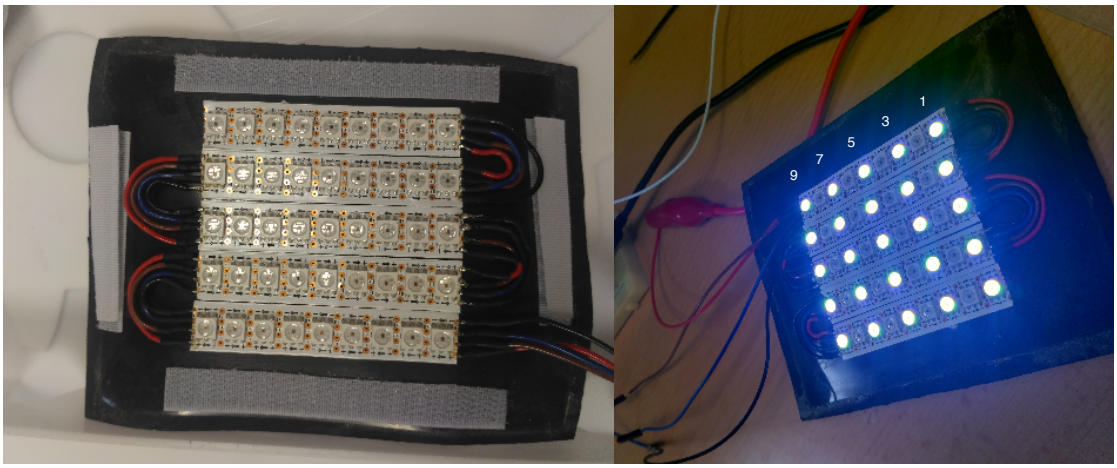

Figura 4.4: Implementación de la matriz LED.

Como se observa en la figura, se han dispuestos las tiras de tal forma que los LEDs queden equiespaciados los unos de los otros. Además, para poder tener una configuración más compacta, las tiras contiguas están giradas 180 grados, tal y como se observa en la Figura 4.5. Si se estira la matriz se puede observar que se trata de la tira original, por lo que la información en una tira viaja en sentido contrario al movimiento en las tiras contiguas como se aprecia en la Figura.

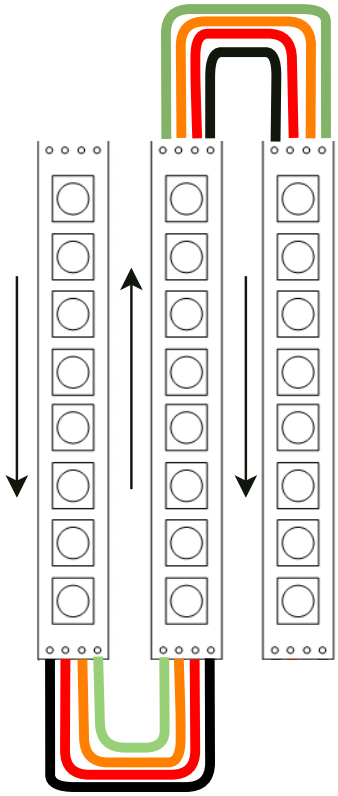

Figura 4.5: Disposición de las tiras LED.

Por otro lado, el transmisor LED se acopla en una camisa. Esto tiene como consecuencia que es necesario que el sistema cuente con una protección en la parte trasera que estará en contacto con la piel. Por otro lado, ya que los LEDs emiten calor hacia atrás, también es insertar un material disipador de calor entre la fuenta LED y la protección. En la Figura 4.4 se observa la fuente LED y los siguientes materiales:

- Manta isotérmica plástica colocada en la parte trasera de la tira LED, como disipador de calor.

- Goma eva colocada en la parte trasera de la manta isotérmica. Tiene como función aislar de una forma cómoda el transmisor LED del cuerpo humano. Destacar que la cómodidad también se busca en los dispositivos *wearables*.

Además, como la fuente LED debe estar lo más pegada posible a la tela de la camisa, se usa velcro de doble cara entre la parte delantera del transmisor LED y la camisa.

Finalmente, en la Figura 4.6 se observa el resultado final de la camisa.

Figura 4.6: Resultado final.

4.1.1.3. Consideraciones del consumo

En la implementación se usa la tira LED de 144LED/m. Esta presenta un consumo máximo de 43.2W/m. Por lo que se obtiene un consumo por LED de 0.3W. En nuestro caso, al tener una matriz de 25 LEDs útiles, esta supone un consumo en potencia de 7.5W. Esto implica que a 5V existe una intensidad de 0.72A. Destacar que estos valores son los valores máximos, cuando se transmite blanco en todos los LED. Sin embargo, en las condiciones de trabajo, la fuente LED trabaja a un 20 % de la capacidad total, esto es, picos de corriente de 0.15 A.

Para reducir la potencia consumida, la fuente LED no permanecerá encendida durante todo el tiempo, sino cuando se desea realizar una transmisión, permaneciendo el sistema en *stand by* hasta que surja la necesidad de transmitir datos.

4.1.2. NUCLEO-L432KC

El otro bloque fundamental en el transmisor es el NUCLEO-L432KC. Este dispositivo es una placa controladora que integra un microcontrolador STM32 de muy baja

potencia (ultra-low-power) basado en un Arm Cortex-M4 de 32 bits que opera a 80 MHz.

En este microcontrolador se implementan tanto el sistema de control para las interfaces de entradas y salidas y el generador de datos aleatorios, como el codificador y el modulador.

4.1.2.1. Entorno de desarrollo

El entorno de desarrollo utilizado en la implementación del NUCLEO-L432KC es Mbed IDE, una plataforma de código abierto desarrollado por ARM, con especial interés en facilitar la implementación de productos que utilizan ARM por distintos fabricantes. El entorno de desarrollo utiliza librerías basadas en los lenguajes de programación C y C++.

Mbed hace uso de una Capa de Abstracción Hardware (HAL) que permite controlar las partes más comunes de los microcontroladores ARM, aportando APIs que facilitan el desarrolo de las aplicaciones. Gracias a estas APIs y al HAL, un mismo programa puede servir para varios microcontroladores que tengan una arquitectura similar.

Las APIs soportadas por Mbed OS se clasifican en:

- APIs de plataforma, para el manejo de propósito general del microcontrolador.
- APIs de drivers, para el manejo de las entradas y salidas analógicas y digitales del microcontrolador.
- APIs RTOS, para el manejo de hilos, sincronización y temporizadores.
- APIs de USB, para el manejo específico del periférico USB.
- APIs de sockets de red, para el manejo de sockets IP. Soporta los protocolos TCP y UDP.
- APIs de interfaces de red, para el manejo de la conectividad del microcontrolador como ethernet, Wi-Fi o telefonía celular.
- APIs de Bluetooth de Baja Energía (BLE).
- APIs de LoraWAN.

- NFC APIs.
- API de seguridad.
- API de almacenamiento.

En esta implementación se hace uso de la APIs de drivers para el manejo de la interfaz SPI y el Ticker, y la API de plataforma o propósito general.

Finalmente, cabe señalar que Mbed ofrece un compilador online, usado en el desarrollo del producto . Al compilar el programa, donde se selecciona el microcontrolador programado, se genera un archivo .bin. Para programar el STM32, solo es necesario conectar el microcontrolador al PC mediante un USB, y copiar el archivo generado directamente en el microcontrolador.

4.1.2.2. Firmware del transmisor

El diagrama de flujo del transmisor se muestra en la Figura 4.7.

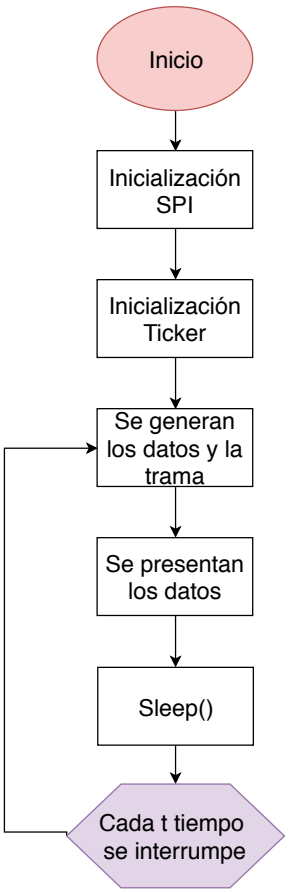

Figura 4.7: Diagrama de flujo del transmisor.

Este diagrama contempla las siguientes tareas durante su ejecución:

1. Inicializa la interfaz SPI para la transmición de los datos a la matriz LED.
2. Inicializa la interrupción a través de la clase Ticker, para la sincronización del envío de datos.
3. Se obtienen los datos a transmitir.
4. Genera la trama a partir de los datos según la tasa de Nyquist configurada para la transmisión.
5. Escribe la trama en la interfaz SPI.
6. Cuando ocurre la interrupción temporal, que coincide con el tiempo de conmutación se levanta, se retorna al paso 3.

SPI

Para la inicialización de la interfaz se usa la clase SPI de la API de Drivers. Para ello se han de especificar los pines correspondientes a las señales MOSI, MISO y CLK. En este caso los pines A6, A5 y A4 respectivamente, son los que corresponden al SPI1 del NUCLEO-L432KC. El conexionado del microcontrolador con la tira de LEDs se puede observar en la Figura 4.8.

```
1 SPI spi(A6, A5, A4);        // MOSI (DATA), MISO (no se usa) y CLK
```

Código 1: Instanciación de la clase SPI.

Una vez instanciado el objeto de la clase, para la inicialización del SPI se ha de especificar el formato mediante la función `spi.format()` y la frecuencia de la señal CLK mediante la función `spi.frecuency()`.

La función `spi.format()` recibe como parámetros el tamaño en bits de la escritura y el modo. El modo en este caso es el 0, que nos indica que el dato se escribe con el flanco de subida del reloj (Polarización = 0) y en la mitad del dato (Fase = 0).

La función `spi.frecuency()` recibe como parámetro la frecuencia de la señal de reloj CLK. El valor elegido es 4000000 correspondiente a una frecuencia de 4MHz.

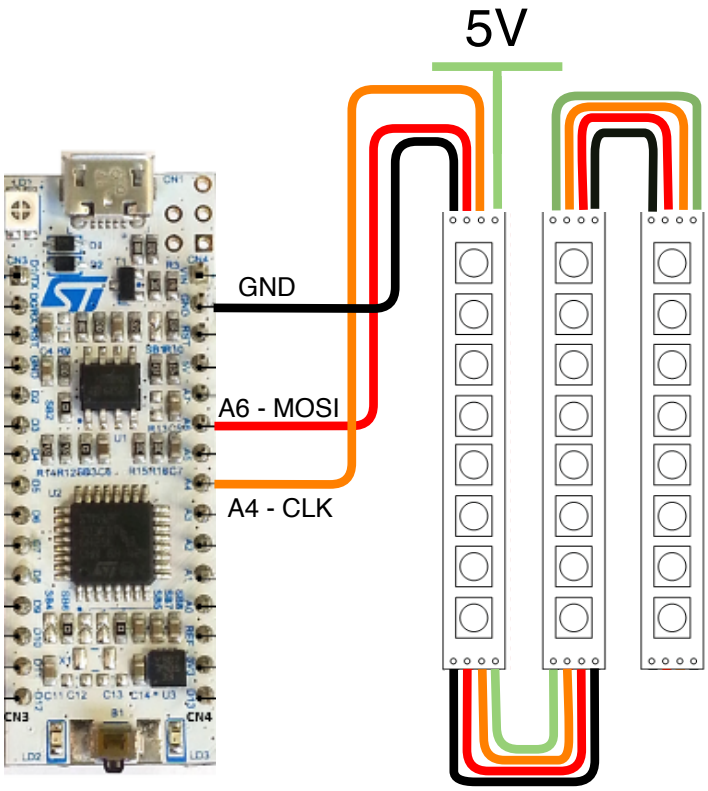

Figura 4.8: Conexionado SPI.

```
1  int main(){
2      ...
3      spi.format(16,0);        // 16 bits, Polarización = 0 y Fase = 0
4      spi.frequency(4000000); // 4MHz
5      ...
6      while(1){
7          ...
8      }
9  }
```

Código 2: Inicialización de la clase SPI.

Ticker

La clase `Ticker` de la API de Drivers de Mbed nos permite realizar una interrupción en el flujo del programa cada cierto tiempo. Cuando esta interrupción temporal sucede, el microcontrolador atiende a una rutina específica declarada en una función al inicio del código. Para hacer uso de esta interrupción es necesario instanciar el objeto `Ticker`

y, posteriormente, iniciarlo mediante la función `ticker.attach_us()` que recibe por parámetros el intervalo de tiempo deseado (en microsegundos) y la dirección a memoria de la función (previamente declarada). La función debe realizar una acción de poco coste computacional. En este caso, se encarga exclusivamente de modificar un indicador, de tal forma que cuando se retorna al bucle principal, el flujo del programa tenga en cuenta que ha sucedido dicha interrupción.

Por otro lado, la función `sleep()` de Mbed nos ofrece la posibilidad de provocar un estado de operación mínima al controlador. Cuando esta función se ejecuta, se bloquea la ejecución del bucle principal, y se pasa a un estado de mínimo consumo. Cuando sucede una interrupción temporal (en nuestro caso), el sistema se despierta, atiende a la interrupción, y continua con la ejecución del programa hasta que encuentra otra instrucción de `sleep()`.

Haciendo uso del `Ticker` y el `sleep()` se puede controlar temporalmente el flujo de la transmisión, dado que los tiempos de interrupción manejados son relativamente altos en comparación con la frecuencia de reloj del microcontrolador. Esto nos proporciona el control de la tasa de Nyquist del transmisor. Dicho tiempo corresponde a t_N/FPS, donde t_N es la tasa de Nyquist y FPS la tasa de captura de frames del receptor.

```
1  Ticker ticker;
2  int main(){
3      ...
4      ticker.attach_us(&funcion_ticker, tiempo_transmision); // t_N/FPS
5      ...
6      while(1){
7          ...
8          sleep();    // se duerme hasta que ocurre una nueva
               ↪  introducción
9      }
10 }
```

Código 3: Uso de la clase Ticker.

Generación de la trama

La trama mencionada en esta sección se puede generar de dos formas distintas como se comenta en el diseño. La primera forma es leyendo una Interfaz de Entrada. La otra forma es generando datos pseudo-aleatorios.

Para la obtención de datos a través de una interfaz de entrada, se puede utilizar la clase AnalogIn de MbedOS para leer los datos de un ADC, o la clase SPI para leer datos de otro dispositivo. Este caso es de libre configuración, depende de la aplicación.

Para la generación de datos pseudo-aleatorios se puede hacer uso de algoritmos PRNG (Pseudorandom Number Generator). En este caso se define la función `rnd()` de generación de datos aleatorios que devuelve un número entero de 16 bits.

Para la generar la trama se utiliza la función `generar_trama_aleatoria()` que recibe por parámetros el vector donde se genera la trama.

```
1  int[25][4] generar_trama_aleatoria(int vector_leds[25][4]){
2      for(int l=0; l<26; l++){
3          if(l==0){
4              vector_leds[l][1] = 255;
5              vector_leds[l][3] = 255;
6          }else if(l==4){
7              vector_leds[l][1] = 255;
8              vector_leds[l][3] = 0;
9          }else if(l==20){
10             vector_leds[l][1] = 0;
11             vector_leds[l][3] = 255;
12         }else if(l==24){
13         }else{
14             random_num = rnd()%14;
15             vector_leds[l][1] = ((random_num&(1<<(i*2)))>>(i*2))*255;
16             vector_leds[l][3] =
                   ↪  ((random_num&(1<<(i*2+1)))>>(i*2+1))*255;
17         }
18     }
19     return vector_leds;
20 }
```

En la línea 1 se obtiene el vector donde se introducen los valores a transmitir. La dimensión del vector es de dimensión 25x4, debido al número de LEDs (25) y a los valores de brillo, canal azul, canal verde y canal rojo necesarios para la configuración de cada LED (4). En las líneas 3, 6, 9 y 12 se especifican los canales rojo y azul de los LED de ancla y entrenamiento. En las líneas 13-17 se genera la información pseudo-aleatoria para enviar. El dato se obtiene utilizando la función `rnd()` y, posteriormente,

se almacena parte del dato en los canales azul y rojo.

Se puede observar que el brillo no se varía en la función, por lo que al instanciar el vector es necesario especificar estos valores. Para los LEDs de datos se especifica un valor de brillo de 3. Para los LEDs de entrenamiento y ancla se especifica un valor de brillo de 10. La diferencia de los valores se midió de forma experimental de tal forma que se pudieran observar las diferencias entre los tipos de LEDs, sin que llegara a producirse la saturación en el receptor. Además, estos valores son los más bajos (el rango de brillo es de 0 a 31).

Presentación de la trama

Para la presentación de la trama se utiliza una función que toma como entrada el vector de la trama a transmitir.

Como se indica en el apartado de la realización física de la fuente LED de este capítulo, para no realizar un número elevado de soldaduras, la tira se dividió en 5 tiras de 9 LEDs cada una. Y ya que cada tira debe poseer 5 LEDs encendidos, esta etapa es la encargada de enviar los campos de configuración de LED vacío para los LEDs intermedios.

Cabe destacar que existen 5 tiras de 9 LEDs cada una, eso hace un total de 45 LEDs, aunque solo 25 sean útiles. Además, en cada posición o LED se tiene la información de brillo, canal verde, canal rojo y canal azul. El codificador y modulador en base a estas consideraciones se encargará de generar una matriz de datos de dimensión [45][4] y presentarla. Para ello se utiliza la función `presentar_strip()` . Esta toma la matriz de datos generada y la transmite a través de la interfaz SPI a la matriz LED.

```
1  void presentar_strip(int vector_leds[25][4]){// Brillo, Rojo, Verde y
   ↪  Azul
2
3      int aux;
4      int posicion;
5
6      spi.write(0x00);  // Campo start (32 bits a 0)
7      spi.write(0x00);
8      spi.write(0x00);
9      spi.write(0x00);
10
```

```
11      for(int l=0; l<5; l++) {
12          if(l%2 == 0){
13              for (int k=0; k<9; k++){
14                  if(k%2==0){ //En las posiciones pares y 0 se envían
                    ↪  los datos.
15                      posicion=(l*10+k)/2;
16                      spi.write((7<<5)|(vector_leds[posicion][0]<<0));
                        ↪   // Campo de datos
17                      spi.write(vector_leds[posicion][1]&0xFF);
                        ↪   // Canal azul
18                      spi.write(1&0xFF);
                        ↪   // Canal verde ON
19                      spi.write(vector_leds[posicion][3]&0xFF);
                        ↪   // Canal rojo
20                  }else{      //En los LEDs intermedios no se envía
                    ↪  información
21                      spi.write(0xFF);
22                      spi.write(0);
23                      spi.write(0);
24                      spi.write(0);
25                  }
26              }
27          }else{
28              aux = 4;
29              for (int k=0; k<9; k++){
30                  if(k%2==0){
31                      posicion = ((l*10+k)/2)+aux;
32                      spi.write((7<<5)|(vector_leds[posicion][0]<<0));
                        ↪  // Trama de datos
33                      spi.write(vector_leds[posicion][1]&0xFF);
34                      spi.write(1&0xFF);
35                      spi.write(vector_leds[posicion][3]&0xFF);
36                      aux = aux-2;
37                  }else{
38                      spi.write(0xFF);
39                      spi.write(0);
40                      spi.write(0);
41                      spi.write(0);
42                  }
43              }
44          }
45      }
```

```
46
47      spi.write(0xFF); // Campo stop (32 bits a 1)
48      spi.write(0xFF);
49      spi.write(0xFF);
50      spi.write(0xFF);
51  }
```

Código 4: Generación de la trama.

En las líneas 6-9 el campo de start se envía a través de la interfaz SPI. En la línea 11 se observa cómo se genera las distintas filas (5) de la matriz. En la línea 12 se analiza si es o no una tira en una posición par o impar (recordar que en la Figura 4.5 se observa que tiras contiguas tienen orientaciones distintas, por lo que la información se debe especificar de forma distinta). En caso de ser una tira par (línea 12) y el LED se encuentra en una posición cero o par (línea 14), se envía el dato (líneas 15-19); en caso contrario, no se envía información de color (líneas 21-24). En caso de ser una tira impar (línea 27) se realiza el mismo prodecimiento, pero en la transmisión del dato por la interfaz SPI, dado que la tira está rotada, la información del último dato se envía en la primera posición.

4.2. Receptor

En este apartado se recoge la implementación del receptor diseñado. Recordar que este sistema implementa la obtención del frame, la adquisición de la señal, y la presentación.

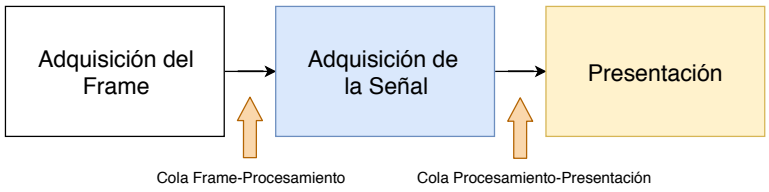

Figura 4.9: Esquema general de la implementación del receptor.

Para la implementación del programa se hace uso del lenguaje de programación orientado a objetos Python. Se usa este lenguaje de programación por su fácil manejo, debido a que es de tipado débil entre otros factores, y facilita el prototipado de las aplicaciones. Aunque el motivo principal de su uso es que librería de OpenCV ofrece soporte para el lenguaje.

OpenCV es una librería de visión artificial de código libre multiplataforma desarrollada por Intel. Esta librería implementa desde algoritmo de obtención de imágenes, operaciones básicas para el filtrado y preprocesamiento de la imagen, hasta el reconocimiento de objetos, y algoritmos de inteligencia artificial. Por esto, dado el carácter del problema atacado, se hace uso de OpenCV.

Esquema multiproceso

La arquitectura general del sistema está basada en técnicas de multihilo o multiproceso. Esto es dado que las peticiones de obtención de frame por parte de la cámara y su presentación son procedimientos bloqueantes. Se entiende procedimientos bloqueantes como aquellos que interrumpen el transcurso del proceso en el que se ejecuta el programa hasta que no hayan finalizado. Por ejemplo, en OpenCV, las llamadas a la función `cv2.imread()` bloquean el programa hasta que no se termine leer la imagen o *frame* desde el stream de datos. También en la presentación, la llamada a la función `cv2.show()` y `cv2.waitKey()` también bloquean el programa durante un tiempo establecido por el `waitKey()`.

Como solución, la estrategia adoptada generalmente consiste en el uso de hilos independientes que se encargan de obtener el frame y almacenarlo en una cola, para que los hilos de procesamiento se encarguen de procesarlo mientras se espera a que llegue el siguiente frame. Sin embargo, se debe considerar el Global Interpreter Lock de Python. El GIL es el mecanismo de Python por el cual se evita que múltiples hilos adquieran el intérprete de Python al mismo tiempo.

Por este motivo, se utiliza un esquema basado en multiprocesamiento. La librería multiprocessing de Python ofrece los recursos asociados a esta funcionalidad. En este caso, se generan tres procesos distintos. Un proceso para cada subsistema: la obtención del *frame*, el descubrimiento y seguimiento, y la presentación.

Esta solución, el multiprocesamiento, presenta un problema, y es el intercambio de datos entre procesos. El problema se puede entender con el problema de la cena de los filósofos, donde en una mesa de N filósofos y cubiertos, cada filósofo necesita 2 cubiertos para comer. Por tanto, es necesario una técnica que indique qué filósofo come primero y quién después. Además, si un filósofo tarda mucho en comer, existirán otro u otros filósofos que tendrán que esperar un tiempo extra. Este problema de

sincronismo también está presente en el multiprocesamiento ya que dos o más procesos quieren acceder al mismo objeto almacenado en memoria. Los mecanismos de control de acceso a memoria son los monitores y los semáforos, entre otros. Estos mecanismos de acceso ya son implementados en el módulo multiprocessing de Python internamente.

Para compartir datos entre procesos de forma fiable y concurrentemente, se hace uso de la clase Queue del módulo multiprocessing de Python.

En nuestro programa se tienen dos colas distintas:

- **Cola Frame-Procesamiento**: en esta cola el subsistema de obtención del frame almacena el *frame* obtenido de la captura de vídeo o cámara, y el subsistema de adquisición de señal toma el *frame* para procesarlo.
- **Cola Procesamiento-Presentación**: en esta cola el subsistema de adquisición de señal pone en la cola el *frame* analizado y la ROI resultante si existe, para que el subsistema de presentación los presente.

La cola implementada es de tipo FIFO, el primer dato que entra es el primer dato saliente. Para la obtención de un dato de la cola se hace uso de la función `get()` y para poner un dato en la cola se hace uso de `put()` . Estas funciones reciben por parámetro cualquier instancia de objeto que se desee almacenar (ventajas de los lenguajes no tipados) y un indicador de si la operación debe realizarse de manera bloqueante o no. En el caso de que se decida actuar de manera no bloqueante se levanta una excepción de tipo cola vacía (`Queue.Empty`) o cola llena (`Queue.Full`) respectivamente, las cuales deben ser tratadas correctamente.

En el programa principal, se levantan todos estos procesos a través de las clases generadas para cada subsistema. El funcionamiento del programa principal sigue las siguientes fases:

- Se instancian las colas.
- Se instancian las clases para cada subsistema. Cada objeto recibe por parámetros las colas necesarias, y los parámetros necesarios para su configuración. Señalar que subsistema de adquisición de señal necesita el acceso a las dos colas.
- Se generan los procesos hijos de los subsistemas adquisición del frame y la presentación, y se inician. El subsistema de adquisición de señal ocurrirá en el proceso

principal, sin necesidad de levantar ningún proceso específico ya que se gobierna el flujo del programa.

- Se inicia la adquisición de la señal, la cual se apoya en las clases descubrimiento y seguimiento.
- Finalmente, una vez finalizada la obtención de frames, se paran los procesos generados y se bloquea al programa principal con la función `join()`, hasta que los procesos hayan finalizado correctamente, y sea posible finalizar correctamente la ejecución del programa.

Todo esto queda recogido en el siguiente fragmento del código:

```
if __name__ == '__main__':

    nombredelvideo_o_camara = ...

    # Se generan las colas
    cola_frame_proc = multiprocessing.Queue()
    cola_proc_pres = multiprocessing.Queue()

    # Se instancian las clases
    cl = AdquisicionFrame(cola_frame_proc)
    ds = AdquisicionSignal(cola_frame_proc, cola_proc_pres,
    ↪  tipoTracker=...)
    sl = Presentacion(cola_proc_pres)

    # Se genera el proceso hijo de la adquisición del frame
    cam_process = multiprocessing.Process(target=cl.cam_loop,
    ↪  args=(nombredelvideo_o_camara, ))
    cam_process.start()

    # Se genera el proceso hijo de la presentación
    show_process = multiprocessing.Process(target=sl.show_loop)
    show_process.start()

    # Se ejecuta la adquisición de la Señal
    ds.run()

    # Se finalizan los procesos hijos
    cam_process.join()
```

```
27      show_process.join()
```

Código 5: Multiprocesamiento.

En la línea 3 se especifica el nombre del vídeo o la cámara a analizar.En las líneas 6 y 7 se generan las colas Frame-Procesamiento y Procesamiento-Presentación, llamadas cola_frame_proc y cola_proc_pres respectivamente. En las líneas 10, 11 y 12 se instancian las clases de los sistemas de adquisición de frame, adquisición de la señal y presentación. En las líneas 15 y 16 se genera e inicia el proceso hijo la adquisición del frame. En las líneas 19 y 20 se genera e inicia el proceso hijo de la presentación. En la línea 23 se utiliza el proceso principal para iniciar la adquisición de la señal. Finalmente, cuando la etapa de adquisición de la señal termina, debido a que no existen frames a analizar, se finalizan los procesos hijos.

A continuación, se explicará la implementación de cada subsistema.

4.2.1. Obtención del frame

En la obtención del frame se obtiene el frame desde un vídeo o una cámara. En ambos casos, la cámara que capta la información debe permitir bajar el tiempo de exposición como se indica en el capítulo del diseño.

Para ello, se implementa la clase `AdquisicionFrame` . Al instanciar un objeto de esta clase, se purga un número determinado de frames para no analizar los *frames* iniciales en los que normalmente se produce la adaptación de la cámara a la configuración establecida Seguidamente, dentro de un bucle (línea 20), se leen constante los frames de la cámara o el vídeo (línea 21) y se guardan en la cola Frame-Procesamiento (línea 26). En caso de no obtener una imagen válida, se sale del bucle principal y se indica la finalización del programa guardando un indicador `False` para la obtención del frame en la cola.

La clase implementada tiene la siguiente forma:

```
1  class AdquisicionFrame:
2      def __init__(self,the_q):
3          self.the_q = the_q              # La cola Frame-Procesamiento
4
5      def purga(self, cap, numFrames):    # Función purga de los N
       ↪  primeros frames
```

```
6           count_frame = 0
7           while count_frame < numFrames:
8               ret, img = cap.read()
9               count_frame += 1
10
11      def cam_loop(self, nombreArchivo):
12
13          # Se inicia la obtención de los frames
14          cap = cv2.VideoCapture(nombreArchivo)
15
16          # Se purga los N primeros frames
17          self.purga(cap, N)
18
19          # Se guardan los siguientes frames en la cola
20          while True:
21              ret, img = cap.read()
22              if ret:
23                  if img is not None:
24                      self.the_q.put_nowait((True,img))
25              else:
26                  self.the_q.put((False,0))   # En caso de no encontrar
                    ↪ la imagen, la imagen guardada es un valor
                    ↪ constante 0.
27                  break
```

Código 6: Clase adquisición de frames.

4.2.2. Adquisición de la señal

La adquisición de la señal es la encargada de procesar los *frames* del vídeo para obtener, finalmente, los datos transmitidos. En este caso, ya que no se implementa ni el decodificador, ni el demodulador, este subsistema sólo se encarga de localizar la ROI que contiene al transmisor LED. La ROI obtenida es almacenada en la cola Procesamiento-Presentación para su posterior presentación por pantalla. Indicar que esta estructura base permite ser modificada para añadir posteriormente las etapas de demodulación y decodificación con facilidad.

Tal y como comenta en el capítulo 3, este subsistema se divide, a su vez, en dos bloques: el descubrimiento y el seguimiento. El descubrimiento se ocupa de localizar la fuente dentro de la imagen generando la ROI descubrimiento. Y el seguimiento se

preocupa de seguir la fuente transmisor a través de los siguientes *frames*, generando la ROI seguimiento.

Las interacciones entre estos bloques se pueden observar en la Figura 4.10, la cual representa el diagrama de estados del subsistema completo. Destacar que las flechas indican la evolución del estado del sistema, donde la parte superior representa la condición, y la parte inferior los mensajes transmitidos entre estados. El `flag_detectado` es un indicador que se utiliza cuando la fuente ha sido localizada en la imagen. El `flag_encontrado` es el que se utiliza cuando la fuente ha sido seguida correctamente. Los saltos entre estados ocurren tras la obtención de un nuevo frame y obtener los flags correspondientes. Entre los saltos del descubrimiento al seguimiento existe una función especial que se encarga de ajusta la ROI descubrimiento al transmisor, generando la ROI seleccionada

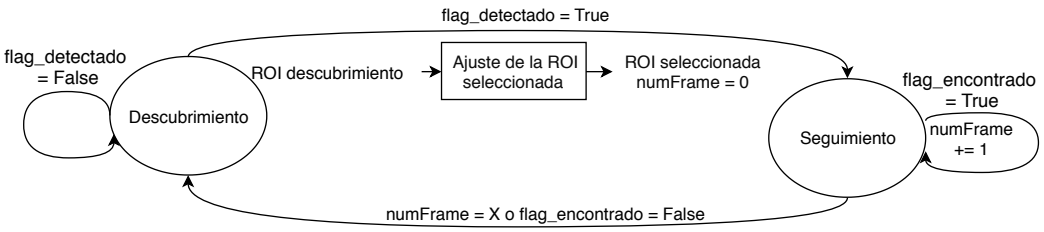

Figura 4.10: Diagrama de estados de la implementación de la adquisición de la señal.

El descubrimiento se sirve de los frames obtenidos en la adquisición del frame mediante la cola Frame-Procesamiento, y descubre si en alguno de ellos existe la fuente transmisora. En el caso de detectar un frame con la fuente transmisora, se genera la ROI descubrimiento generando la ROI seleccionada. Esta ROI es usada para iniciar el seguimiento. Cada cierto tiempo se vuelve a la etapa de descubrimiento. Esto es debido, como se ha comentado, a que la ROI seguimiento se va degenerando a lo largo del tiempo. Mencionar que el tiempo en este caso es medido en número de *frames* procesados. Por esto, al iniciar el seguimiento, la variable `numFrames` toma el valor cero.

Finalmente, es importante remarcar que para el descubrimiento y el seguimiento se hace uso del canal verde, ya que este permanece teóricamente invariante con el tiempo. En la práctica, el canal verde varía muy poco debido a las interferencias de los canales adyacentes, rojo y azul.

A continuación, se detallan la implementación de los bloques utilizados para cada

estado, y la función de adapación de la región seleccionada.

4.2.2.1. Descubrimiento

Como se comenta en el diseño, el descubrimiento tiene dos fases. Estas son la generación de proposals y la clasificación. La relación se expone en la Figura 4.11.

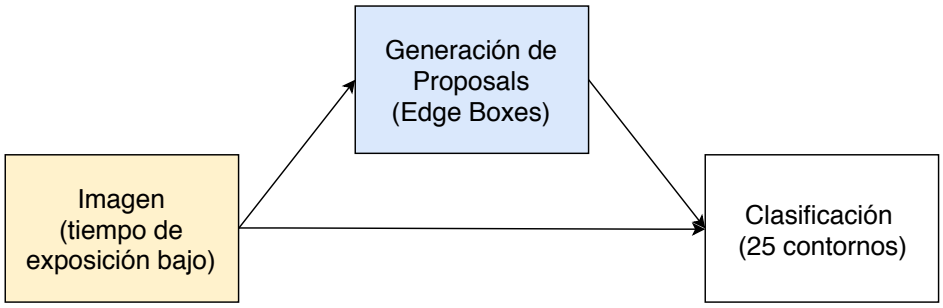

Figura 4.11: Diagrama del descubrimiento.

En la primera fase, al obtener el *frame*, se hace uso de algoritmos de generación de *proposals* sobre la imagen. Los algoritmos implementados para Python por OpenCV en el módulo de ximgproc son el *Selective Search* y el *Edge Boxes*. Se elige el algoritmo de generación de proposals de *Edge Boxes* por las razones expuestas en el capítulo 2.

En la segunda fase, la clasificación, se determina si dentro de cada proposals existen 25 contornos. Se elige este criterio de clasificación dada la geometría del transmisor LED y la naturaleza de la imagen, ya que existen pocos objetos luminosos por si mismos que presenten la misma forma.

Estas fases se ejecutan en cada *frame* hasta obtener una ROI que contenga al transmisor. Una vez obtenido, se pasa al ajuste de la ROI seleccionada y al seguimiento.

Para su implementación en Python, se hace uso de tres clases: `Proposals`, `Clasificacion` y `Descubrimiento`. Como sus nombres indican, la primera clase implementa el algoritmo de detección de proposal *Edge Boxes*; la segunda clase implementa la clasificación de los 25 contornos o LEDs; y finalmente, la clase `Descubrimiento` se encarga de gestionar las entradas y salidas y las interacciones entre las clases anteriores.

Clase Proposals

La estructura de la clase Proposals sigue el siguiente pseudocódigo.

```
1  class Proposals():
2      def __init__(self, model, maxBoxes):
3          self.edge_detection =
           ↪  cv.ximgproc.createStructuredEdgeDetection(model)
4          self.edge_boxes = cv.ximgproc.createEdgeBoxes()
5          self.edge_boxes.setMaxBoxes(maxBoxes)
6
7      def execute(self, im):
8          edges =
           ↪  self.edge_detection.detectEdges(np.float32(cv.cvtColor(im,
           ↪  cv.COLOR_BGR2RGB))/255.0)
9          orimap = self.edge_detection.computeOrientation(edges)
10         edges = self.edge_detection.edgesNms(edges, orimap)
11         proposals = self.edge_boxes.getBoundingBoxes(edges, orimap)
12         return proposals
```

Código 7: Clase para la generación de proposals.

La clase Proposals posee dos métodos: init, donde se inicializa el algoritmo de Edge Boxes mediante las funciones del módulo ximgprob de OpenCV; y execute, donde se ejecuta el algoritmo para las imágenes pasadas como argumentos.

En la línea 2 se observa la función init, donde se pasa por parámetros el modelo y el número máximo de proposals que puede generar el algoritmo. En la línea 3, 4 y 5, `cv.ximgproc.createStructuredEdgeDetection()` genera un objeto con la estructura del algoritmo, `cv.ximgproc.createEdgeBoxes()` se instancia el algoritmo, y `setMaxBoxes()` establece el número máximo de proposals que pueden generar. Estos objetos serán usados en la función execute.

En la línea 7 se observa que la función execute obtiene por parámetros la imagen a ser analizada. En la línea 8 se utiliza la estructura generada para detectar los bordes del macrobloque con la función `detectEdges()` . En la línea 9 se genera el mapa de orientación de los bordes (`computeOrientation()`). En la línea 10 se mejoran los bordes utilizando al mapa de orientaciones (`edgesNms()`). En la línea 11 se generan la lista de proposals utilizando el modelo (`getBoundingBoxes()`). Finalmente, en la línea 12 se retorna la lista generada.

Clase Clasificación

La estructura de la clase Clasificación es:

```
1  class Clasificacion():
2      def __init__(self, threshold_var, threshold_max):
3          ...
4
5      def execute(self, im, proposals):
6
7          flag_encontrado = False
8
9          for roi in proposals:
10
11             im_min = im[(y):(y+h),(x):(x+w),1]
12             ret,thresh = cv2.threshold(im_min,self.threshold_var,self.
               ↪  threshold_max,cv.THRESH_BINARY)
13             contours, hierarchy = cv2.findContours(thresh,
               ↪  cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
14
15             if len(contornos) == 25:
16                 flag_encontrado = True
17                 break
18
19         return flag_encontrado, roi
```

Código 8: Clase Clasificación.

La clase Clasificación posee dos funciones: `init` (el constructor de la clase), donde se establecen los valores de umbralización de la imagen para obtener los contornos; y `execute`, clasifica si dentro de la lista devuelta por la clase `Proposals` existe un proposal que contenga 25 contonos.

En la línea 2 se observa cómo la función `init` recibe por parámetros los parámetros de umbralización de la imagen: `threshold_max`, que indica el valor máximo de la imagen, normalmente 255; y `threshold_var`, que indica el umbral de binarización (40 para este caso). En la umbralización binaria o binarización, si existe algún píxel menor al umbral, su valor se cambiará al mínimo o 0, y para todos los píxeles con valor mayor al umbral, su valor se cambiará al máximo o 255.

En la línea 5 se observa que la función `execute` obtiene por parámetros la imagen y la lista de proposals a evaluar para su clasificación. En la línea 7 se inicializa el flag que indica si se ha descubierto alguna ROI válida. En la línea 9 se observa cómo se itera la lista de proposals. En la línea 11 se obtiene la imagen que contiene cada

proposal. En la línea 12 se umbraliza la imagen. Para la umbralización se usa la función `cv2.threshold()` donde se especifica la imagen, los valores de umbralización y el tipo de binarización. En este caso, el tipo de umbralización es `cv.THRESH_BINARY` que indica que es una umbralización binaria, se obtiene una imagen binarizada a dos valores: 0 y 255.

En la línea 13 se obtienen los contornos mediante la función `cv2.findContours()`. Esta función está basada en el análisis de los bordes de las imágenes binarizadas. Particularmente, en el análisis topológico de las estructuras de los bordes. El algoritmo hace uso de la relación de bordes externos e internos (un ejemplo es un donut donde la figura viene contenida entre los dos bordes). Para objetos sin huecos, solo se observan los bordes superiores. Esta técnica es útil para contabilizar contornos, base fundamental de este trabajo.

En la línea 15 se comprueba si el número de contornos es igual a 25. En caso afirmativo, se cambia el flag de encontrado a True, y se sale del bucle. Finalmente, se retorna el flag y la ROI. En caso de que el flag sea False, la ROI devuelta no se corresponde a ninguna ROI válida.

Clase Descubrimiento

La estructura de la clase descubrimiento se muestra en el siguiente pseudocódigo:

```
1  class Descubrimiento():
2      def __init__(self, model, maxBoxes, threshold_var, threshold_max):
3          self.boundingBoxes_EdgeBoxes = Proposals(model, maxBoxes)
4          self.clasificacion = Clasificacion(threshold_var,
           ↪  threshold_max)
5
6      def execute(self,im):
7          proposals = self.proposals.execute(im)
8
9          if len(proposals) != 0:
10             flag_encontrado, roi = self.clasificacion.execute(im,
               ↪  proposals)
11
12             if encontrado:
13                 flag_descubrimiento = True
14                 roi_descubrimiento = roi
```

```
15              else:
16                  flag_descubrimiento = False
17                  roi_descubrimiento = None
18
19          else
20              flag_descubrimiento = False
21              roi_descubrimiento = None
22
23          return flag_descubrimiento, roi_descubrimiento
```

Código 9: Clase Descubrimiento.

La clase Descubrimiento posee dos funciones: `init` , donde se instancian las clases Proposals y Clasificacion; y `execute` , donde a partir de una imagen, se obtiene la ROI descubrimiento. Mencionar que en esta clase coordina las relaciones entre las clases Proposals y Clasificacion como se indica en el inicio.

En la línea 2 se obtienen los parámetros necesarios para instanciar en las líneas 3 y 4, las clases Proposals y Clasificacion.

En la línea 6 se obtiene por parámetro la imagen a clasificar. En la línea 7 se obtiene la lista de proposals. En el caso de que la lista contenga algún *proposal*, es decir, que su longitud sea distinta a cero, se analiza si existe algún proposal que contenga 25 contornos. En caso negativo, el flag que indica que se ha descubierto se pone a `False` y la ROI descubrimiento a `None` como observa en las líneas 20 y 21. En la línea 10 se devuelve el indicador de quese ha encontrado una fuente y su localización en la imagen.

Finalmente, en la línea 23 se retorna el `flag_descubrimiento` y la ROI descubrimiento.

4.2.2.2. Ajuste de la ROI seleccionada

Esta parte es la encargada de ajustar la ROI seleccionada al transmisor LED. Para ello, se implementa la función adjusteRoiseleccionada que tiene la siguiente estructura:

```
1  def adjusteRoiseleccionada(delta, roi_descubrimiento, im)
2          x,y,h,w = roi_descubrimiento
3          im_min = im[(y):(y+h),(x):(x+w),1]
4          ret,thresh = cv2.threshold(im_min,self.threshold_var,self.thre⌋
           ↪  shold_max,cv.THRESH_BINARY)
```

```
5           contours, hierarchy = cv2.findContours(thresh, cv.RETR_TREE,
            ↪  cv.CHAIN_APPROX_SIMPLE)
6
7           left = 0; right = 0; up = 0; down = 0
8           for cnt in contours:    # Se miran en todos los contornos.
9               for cnti in cnt:    # Los contornos son una lista de
                ↪  puntos, se mira en todos los puntos del contorno.
10
11                  x_cnti,y_cnti = cnti[0]
12
13                  if x_cnti>right:
14                      right = x_cnti
15                  elif x_cnti<left:
16                      left = x_cnti
17                  else:
18                      pass
19
20                  if y_cnti>up:
21                      up = y_cnti
22                  elif y_cnti<down:
23                      down = y_cnti
24                  else:
25                      pass
26
27          roi_seleccionada = (x+left), (y+down), (right-left), (up-down)
28
29          roi_seleccionada = (roi_seleccionada[0]-(delta/2)),
            ↪  (roi_seleccionada[1]-(delta/2)),
            ↪  (roi_seleccionada[2]+delta), (roi_seleccionada[3]+delta)
30
31          return roi_seleccionada
```

Código 10: Adaptación de la ROI seleccionada.

Esta función es la encargada de calcular la ROI seleccionada. Para ello, se generan todos los contornos dentro de la ROI descubrimiento. Posteriormente, se generan cuatro variables que representa el valor límite en cada sentido. En cada contorno de la lista de contornos, se observa si el contorno presenta un valor menor (en caso de izquierda y abajo) o mayor (en caso de arriba y derecha) al valor anterior. Esto se realiza de forma iterativa para todos los contornos. Finalmente se obtiene la ROI que mejor encierra el transmisor LED. Finalmente, se aplica el margen para generar la ROI seleccionada.

En la línea 1 se observa cómo recibe por parámetros la imagen, la ROI descubrimiento y la delta. Cabe señalar que la delta es el margen que se aplica a la ROI ajustada para generar la ROI seleccionada. En las líneas 2-5 se binariza la imagen dentro de la ROI descubrimiento y se obtienen los contornos. En las líneas 7 se inician los valores de arriba, abajo, izquierda y derecha a cero, para que en las líneas 8-25, de forma iterativa, se calculan los cuatro sentidos límites para los contornos. En la línea 27 se adapta el formato al del ROI. El formato de la ROI es [x, y, ancho, alto] donde el punto (x,y) representa el punto del transmisor más próximo al origen en coordenadas cartesianas. En la línea 29 se aplica el margen a la ROI ajsutada para generar la ROI seleccionada. Finalmente, en la línea 31 se retorna la ROI seleccionada.

4.2.2.3. Seguimiento

Finalmente, cuando se obtiene la ROI seleccionada, se inicia el Seguimiento. Dicha clase tiene tres funciones distintas. La primera es el init de Python donde se especifica el tipo de algoritmo de seguimiento que se va a usar. La segunda es la función `init_tracker()` que inicia el algoritmo toma por parámetros la ROI seleccionada y la imagen, Finalmente, la tercera función `update()` o actualización, donde se pasa una nueva imagen y el algoritmo devuelve una ROI seguimiento donde es muy probable que se encuentre el objeto seguido. La clase se implementa de la siguiente forma:

```
1  class Seguimiento():
2      def __init__(self, tipoTracker):
3          tracker_lista = ['BOOSTING', 'MIL','KCF', 'TLD', 'MEDIANFLOW',
           ↪  'GOTURN', 'MOSSE', 'CSRT']
4          tracker_tipo = tracker_lista[tipoTracker]
5          if tracker_tipo == 'BOOSTING':
6              self.tracker = cv2.TrackerBoosting_create()
7          if tracker_tipo == 'MIL':
8              self.tracker = cv2.TrackerMIL_create()
9          if tracker_tipo == 'KCF':
10             self.tracker = cv2.TrackerKCF_create()
11         if tracker_tipo == 'TLD':
12             self.tracker = cv2.TrackerTLD_create()
13         if tracker_tipo == 'MEDIANFLOW':
14             self.tracker = cv2.TrackerMedianFlow_create()
15         if tracker_tipo == 'GOTURN':
```

```
16              self.tracker = cv2.TrackerGOTURN_create()
17          if tracker_tipo == 'MOSSE':
18              self.tracker = cv2.TrackerMOSSE_create()
19          if tracker_tipo == "CSRT":
20              self.tracker = cv2.TrackerCSRT_create()
21
22      def init_tracker(self, im, roi_seleccionada):
23          return self.tracker.init(im,tuple(roi_seleccionada))
24
25      def update(self,flag,im):
26          if flag:
27              flag_seguimiento, roi_seguimiento =
                ↪  self.tracker.update(im.astype('uint8'))
28              return flag_seguimiento, roi_seguimiento
29
30          else:
31              return False, None
```

Código 11: Clase Seguimiento.

En la línea 2 se pasa por parámetros el índice de la lista `tracker_lista` que especifica el tipo de algoritmo de seguimiento usado. En las líneas 3-20 se observa cómo se instancia el tracker en función del índice de la lista.

En la línea 22 se pasa por parámetros la imagen y la ROI seleccionada necesaria para iniciar el algoritmo de seguimiento. En la línea 23 se observa cómo se inicia utilzando a la función `tracker.init` . Esta función devuelve un flag donde se indica si se ha podido iniciar.

En la línea 25 se pasa por parámetros el flag y la imagen. En caso de que la imagen sea válida o el flag igual a `True` , se actualiza el tracker utilizando la función `tracker.update()` . Esta función recibe por parámetros la imagen, y devuelve la ROI seguimiento y el flag que indica si se ha localizado correctamente dicha ROI. En el caso que el flag de la imagen sea `False` , en la línea 31, la imagen no es válida, y se devuelve un `False` y como ROI un `None` .

4.2.2.4. Adquisición de la señal

Las clases expuestas en las secciones anteriores deben interactuar entre ellas como se indica en la Figura 4.10. Para ello se implementa la clase `AdquisicionSignal` . Recordar

que esta clase es la responsable de obtener el *frame* de la cola Frame-Procesamiento y poner la ROI seguimiento en la cola Procesamiento-Presentación.

La clase se estructura de la siguiente forma:

```
class AdquisicionSignal():
    def __init__(self, queue_frame_proc, queue_proc_presenta,
    ↪ model="model.yml", maxBoxes=..., threshold_var=...,
    ↪ threshold_max=..., tipoTracker=...):
        ...
        self.descubrimiento = Descubrimiento(model, maxBoxes,
        ↪ threshold_var, threshold_max)

    def run(self, delta):
        flag_isRunning = True
        estado = 'descubrimiento'

        while flag_isRunning:

            if estado == 'descubrimiento':
                flag_isRunning, im = self.queue_frame_proc.get()
                flag_detectado, roi_descubrimiento =
                ↪ self.descubrimiento.execute(flag_isRunning, im)

                if flag_detectado:
                    estado = 'seguimiento'
                    roi_seleccionada = ajusteROIseleccionada(delta,
                    ↪ roi_descubrimiento, im)
                    self.queue_proc_presenta.put([im,roi_seleccionada])

                else:
                    self.queue_proc_presenta.put([im,None])

            else:
                if self.seguimiento in locals() or global():
                    pass
                else:
                    self.seguimiento = Seguimiento(self.tipoTracker)
                    flag_tracker = self.seguimiento.init_tracker(im,ro⌋
                    ↪ i_seleccionada)
                    if flag_tracker:
                        pass
                    else:
```

```
33                          numFrames = X - 1
34
35                  flag_isRunning, im = self.queue_frame_proc.get()
36                  numFrames += 1
37
38                  if numFrames = X:
39                      self.queue_proc_pres.put([im,None])
40                      estado = 'descubrimiento'
41                      del tracker
42
43                  else:
44                      flag_encontrado, roi_seguimiento =
                        ↪ self.seguimiento.update(flag_isRunning, im)
45
46                      if flag_encontrado:
47                          if len(contours) == 25:
48                              self.queue_proc_pres.put([im,roi_seguimien
                                ↪ to])
49
50                          else:
51                              self.queue_proc_pres.put([im,None])
52                              estado = 'descubrimiento'
53                              del tracker
54
55                      else:
56                          self.queue_proc_pres.put([im,None])
57                          estado = 'descubrimiento'
58                          del self.seguimiento
59
60          self.queue_proc_pres.put([im,-1])
```

Código 12: Clase AdquisitionSignal.

La clase AdquisicionSignal posee dos funciones: `init`, donde se instancia la clase Descubrimiento; y run, donde se interactúa entre los objetos de las clases Descubrimiento y Seguimiento, y se obtiene la ROI seguimiento que se almacena en la cola Procesamiento-Presentación a partir de la información obtenida de la cola Frame-Presentación.

El funcionamiento de la clase sigue el diagrama de flujo mostrado en la Figura 4.12.

En la línea 4, dentro de la función `init`, se instancia el objeto de la clase Descubrimiento. En la línea 7 y 8, dentro de la `run`, se inician las variables que indican el estado del sistema y el flag que indica que la imagen es válida. Si el estado es "descubrimiento" (línea 12), se descubre si la fuente transmisora está dentro de la imagen. En caso de que la fuente se encuentre en la imagen (línea 16), se cambia el estado y se pasa al seguimiento (línea 17). Además, se ajusta la ROI descubrimiento a la fuente mediante la función `ajusteROIseleccionada`, generando la ROI seleccionada (líneas 17, 18). Una vez en el seguimiento, si el estado anterior del sistema es el descubrimiento, se instancia la clase Seguimiento y se intenta iniciar (líneas 28 y 29). Si no se puede iniciar, se establece el `numFrame` igual a X-1 (línea 33), lo que produce que en la siguiente obtención del *frame* se llegue al límite establecido y se cambie el estado del sistema al descubrimiento. En la línea 35 y 36 se obtiene el frame y se incremenenta en una unidad el contador `numFrame`. En las líneas 38-41, En caso de que se llegue al límite, se cambia el estado a descubrimiento y se borra el objeto de seguimiento. Si no se ha llegado al límite, se obtiene la ROI seguimiento del nuevo *frame* (línea 44). En caso de encontrar el objeto deseado en el seguimiento, se observa si existe la fuente en la ROI (observando el número de contornos). En caso de que no se encuentre la fuente, se cambia el estado y se retorna al descubrimiento. Si existe, se permanece en el seguimiento. Todos los procedimientos descritos para el seguimiento se recogen en las líneas 44-58. Cabe destacar que en cada caso, los datos de imagen y ROI se guardan en la cola Procesamiento-Presentación para que el subsistema de presentación los pueda mostrar.

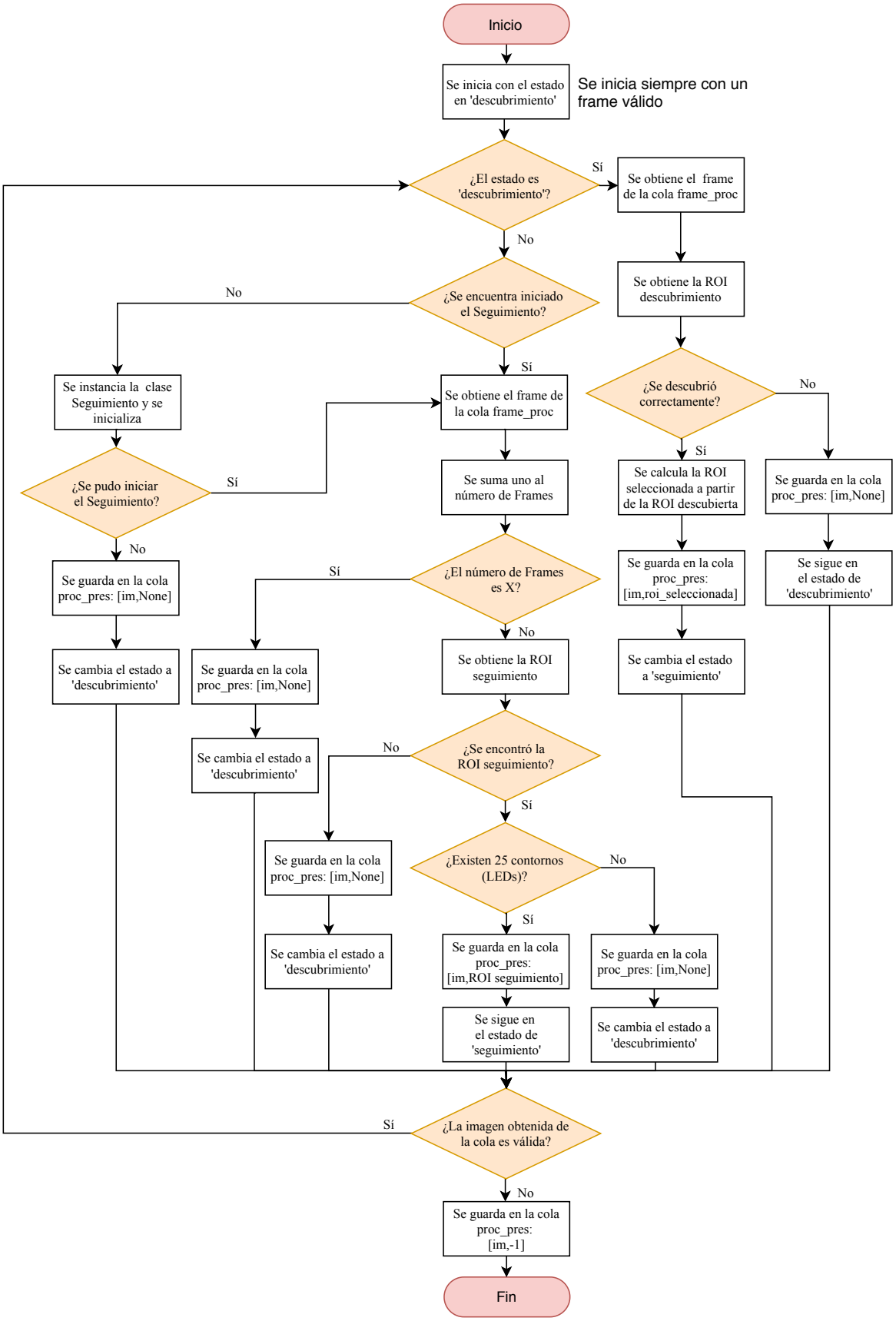

Figura 4.12: Diagrama de flujo de la clase AdquisicionSignal.

4.2.3. Presentación

El subsistema de presentación es el encargado de mostrar los resultados obtenidos del subsistema adquisición de la señal. En caso de obtener alguna ROI donde se encuentre el objeto estudiado, la ROI se dibuja sobre la imagen que se obtiene de la cola. En el caso contrario, solo se presenta el *frame*.

La clase implementada tiene la siguiente estructura:

```
1  class Presentacion:
2      def __init__(self,cola_proc_pres):
3          ...
4
5      def show_loop(self):
6          cv2.namedWindow('salida')
7          roi_seleccionada = 1
8          while roi_seleccionada!= -1:
9              im, roi_seleccionada = self.cola_proc_pres()
10
11             if roi_seleccionada is not None:
12                 if roi_seleccionada == -1:
13                     pass
14                 else:
15                     x,y,w,h = roi_seleccionada
16                     cv2.rectangle(im, (x, y), ((x+w), )y+h)),
                       ↪  (255,0,0), 2)
17                     cv2.imshow('salida', im)
18                     cv2.waitKey(1)
19             else:
20                 cv2.imshow('salida', im)
21                 cv2.waitKey(1)
22         cv2.destroyAllWindows()
```

Código 13: Clase Adquisición de Frames.

La clase Presentacion posee dos funciones: init, donde se toma por parámetros la cola Procesamiento-Presentación; y run, donde se presenta la información obtenida de la cola.

En la línea 2 se obtiene por parámetros la cola Procesamiento-Presentación. En la línea 6 se inicia una ventana con nombre “salida” donde se presentará la información obtenida. En la línea 7 inicia la ROI seleccionada con un valor -1. Recordar que este

valor nos indica que se ha terminado el vídeo o la transmisión de la cámara. En la línea 9 se obtiene la imagen y la ROI seguimiento de la cola. En caso de que la ROI sea None, línea 16, no se obtiene la ROI seguimiento y, por tanto, solo se presenta la imagen, líneas 20-21. La presentación se realiza mediante cv2.imshow y cv2.waitKey. En caso de que la ROI sea -1, línea 19, se sale del bucle while y se cierra la ventana, línea 22. En caso de que no sea None y tenga un valor distintos de -1, línea 11, se obtiene la ROI seguimiento. En este caso, la ROI se dibuja sobre la imagen mediante `cv2.rectangle()` y se presenta por pantalla, líneas 15-18.

Capítulo 5

Evaluación del sistema

Una de las fases más importantes en el desarrollo de un sistema es su evaluación. La evaluación de un sistema puede ser llevada a cabo mediante simulaciones controladas, emulaciones del comportamiento real del sistema, o bien, mediante la ejecución de tests experimentales. En todos los casos es necesario desarrollar un entorno de evaluación. En este caso, se optó por llevar a cabo una implementación física del entorno de evaluación para la realización de una serie de pruebas experimentales.

El desarrollo del entorno de evaluación se divide en varias fases, tal y como se muestra en la Figura 5.1.

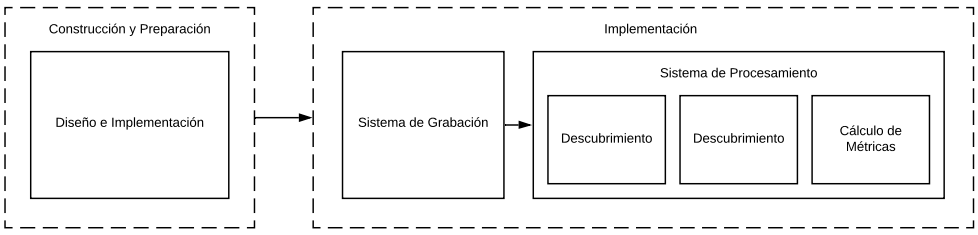

Figura 5.1: Fases de la evaluación.

- **Construcción y preparación**: en esta fase se diseña e implementa el entorno de evaluación. En la sección de Diseño e Implementación, se discuten los distintos elementos que componen el entorno, así como los parámetros a tener en cuenta para la ejecución de las pruebas.

- **Ejecución y bbtención de resultados**: esta fase se realiza en tres partes: la etapa de grabación, la etapa de procesamiento *offline* de los frames de cada vídeo,

y la etapa del cálculo de las métricas de evaluación. La evaluación se divide en dos pruebas:

- **Prueba de descubrimiento**: en la que se evalúa el rendimiento del bloque de descubrimiento. Para ello se obtienen los datos necesarios para calcular la sensibilidad y el tiempo medio de ejecución del algoritmo implementado.
- **Prueba de seguimiento**: en la que se evalúa la calidad del seguimiento. Para ello, se comparan los resultados obtenidos durante la prueba de descubrimiento y se evalúa así el desempeño de los distintos algoritmos de seguimiento disponibles.

En la siguiente sección se discuten las métricas a tener en cuenta durante la realización de ambas pruebas.

5.1. Métricas

Las métricas a estudiar están estrechamente relacionadas con los subsistemas implementados del receptor: el descubrimiento y el seguimiento.

5.1.1. Métricas en la etapa del descubrimiento

El descubrimiento es la etapa responsable de detectar la fuente transmisora dentro de la imagen. Esta etapa supone un costo computacional mayor que la etapa de seguimiento, por lo que uno de los parámetros fundamentales a estudiar es el tiempo de ejecución medio del algoritmo. Por otra parte, también se ha de tener en cuenta la sensibilidad a la hora de detectar la fuente transmisora correctamente.

El parámetro de sensibilidad usado viene indicado en la ecuación:

$$Sensibilidad = \frac{True_{positives}}{(True_{positives} + False_{negatives})} \tag{5.1}$$

Donde los casos de $True_{positives}$ son aquellos casos en los que el receptor detecta una fuente legítima en la imagen correctamente. Los casos designados como $False_{negatives}$, son los casos en los que el sistema considera que hay una fuente cuando no la hay.

Para calcular los casos $True_{positives}$, se cuantifican el número de frames en los cuales se ha encontrado el transmisor, en caso contrario, se toma como $False_{negative}$, ya que,

en los vídeos obtenidos la imagen siempre contendrá la fuente. Por otra parte, cabe destacar que la probabilidad obtenida es una estimación, ya que para asegurar la total fiabilidad de los resultados, se han de tomar infinitas muestras, o un conjunto extenso para su aproximación.

Para la métrica temporal se calcula la media del conjunto de los tiempos obtenidos durante la ejecución del algoritmo para cada *frame* del vídeo.

5.1.2. Métricas de la etapa de seguimiento

El seguimiento, en cambio, se encarga de seguir el transmisor. Para entender las métricas se debe recordar la diferencia entre las ROI empleadas, estas son:

- **ROI descubrimiento**: la ROI que más se ajusta al contenido del transmisor LED.
- **ROI seleccionada**: la ROI que se le indica inicialmente al algoritmo de seguimiento. Es la región obtenida del descubrimiento a la cual se le aplica un margen en base a un porcentaje del ancho que se indica como parámetro.
- **ROI seguimiento**: la ROI que devuelve el algoritmo de seguimiento.

El seguimiento es inicializado con la ROI seleccionada y, en nuevos frames, devuelve la ROI seguimiento. Por esto, es preciso analizar, qué tan ajustada se encuentra esta ROI seguimiento respecto a la ROI descubrimiento. Este ajuste se puede interpretar en términos de excentricidad entre ambas regiones.

La excentricidad cuantifica el hecho de que las ROI del descubrimiento y seguimiento no compartan el mismo centro de masas, es decir, que no estén centradas la una respecto a la otra. En la práctica se desea que la excentricidad sea nula, es decir, que estén centradas y que el ajuste sea bueno.

Para estimar la excentricidad, se ha diseñado un método basado en un conjunto de parámetros K. Estos parámetros representan la distancia superior (K_u), inferior (K_d), izquierda (K_l) y derecha (K_r), entre los segmentos de la ROIs. En la Figura 5.2 se muestra un caso en el cual la ROI descubrimiento se encuentra perfectamente centrada.

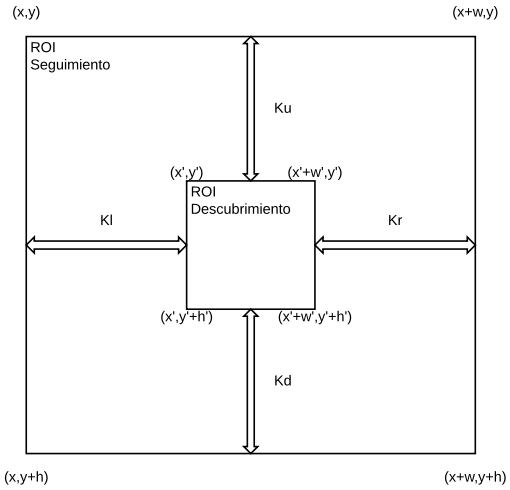

Figura 5.2: Representación parámetros K.

Los parámetros K se definen en la Ecuación 5.2.

$$\begin{aligned} K_u &= y' - y \\ K_d &= (h' - h) - K_u \\ K_l &= x' - x \\ K_r &= (w - w') - K_l \end{aligned} \tag{5.2}$$

El análisis independiente de cada parámetro K, y cómo afecta a la excentricidad es complejo. Es por este motivo, que este análisis se realiza mediante la media y la desviación estándar de estos parámetros.

La media nos indica la relación de escala entre la ROI seguimiento y la ROI descubrimiento. En el caso de tener media nula, nos indica que las ROIs son iguales en área. En caso contrario, una ROI tiene un área mayor a la otra. Lo que se desea es que el incremento en área sea constante e igual al parámetro que se indica en el seguimiento. Además, en caso de tener una media negativa, nos indica que parte del transmisor queda fuera de la ROI. En ese caso, es necesario volver a descubrir el transmisor.

En segundo lugar, la desviación estándar nos indica que tan centrado se encuentran las ROIs. En el caso de tener desviación nula, las distancias en todos los sentidos son parecidas, por lo que se puede señalar que la ROI seguimiento se encuentra perfectamente centrada. En caso contrario, cuanto mayor sea esta desviación, más descentrada estará dicha ROI.

En definitiva, el análisis de los parámetros K contempla los siguientes casos:

- **media nula** las ROI son las mismas.
- **media no nula**: en este caso, se debe tener en cuenta si es o no negativa.
  - **media negativa**: el transmisor LED queda parcial o completamente fuera. Esto implica que se debe volver a la etapa de descubrimiento.
  - **media positiva**: indica que el objeto está dentro de la ROI seguimiento y tiene un área mayor a la real.
- **desviación estándar nula**: el seguimiento está perfectamente centrado respecto al caso real.
- **desviación estándar no nula**: el seguimiento está descentrado.

Evolución temporal de los Parámetros K

Por otro lado, para poder observar la evolución temporal de la media y la desviación estándar de los Parámetros K se hace uso de la Regresión Lineal.

La regresión Lineal es un modelo matemático que modela la relación entre variables de forma lineal. En este caso, ya que se quiere observar la dependencia temporal, se hace la regresión lineal entre el conjunto de la media de los Parámetros K (escala) en cada frame y el conjunto del número de *frames*, y la regresión lineal del conjunto de las desviaciones estándar (centralidad) en cada frame y el conjunto de todos los *frames*.

Ya que la Regresión Lineal modela la relación como una recta, esta queda bien definida con la pendiente β_1 y el punto inicial β_0. En este caso, Y hace referencia a la media o desviación estándar y X al número de frame. Por tanto, para una X_i determinada, se tiene una $Y_{estimada_i}$ que viene definida en la Ecuación 5.3.

$$Y_{estimada_i} = \beta_0 + \beta_1 * X_i \tag{5.3}$$

Realmente, la $Y_{estimada_i}$ difiere de la Y_{real_i} un valor e_i o valor residual. Se debe cumplir que la distribución del valor residual siga una distribución gaussiana de media nula y varianza no nula para poder aplicar la regresión lineal. La Y_{real_i} queda descrita en la Ecuación 5.4.

$$Y_{real_i} = \beta_0 + \beta_1 * X_i + e_i \tag{5.4}$$

Para poder obtener los valores de β_0 y β_1 que mejor se adapten a la Y_{real_i}, se debe buscar un mínimo para los valores residuales e_i. Destacar que los valores residuales se pueden obtener mediante la resta de la estimada y la real.

$$e_i = Y_{real_i} - Y_{estimada_i} \tag{5.5}$$

Para buscar el mínimo, se aplica la técnica de mínimos cuadrados. En esta técnica se realiza la suma cuadrática de los valores residuales y se computan las derivadas parciales respecto a los parámetros β_0 y β_1. El mínimo se obtiene al igualar las derivadas a cero, tal y como se muestra en el sistema de Ecuaciones 5.6.

$$\begin{aligned} \frac{\partial \sum_{k=1}^{N} (Y_{real_i} - Y_{estimada_i})^2}{\partial \beta_0} &= 0 \\ \frac{\partial \sum_{k=1}^{N} (Y_{real_i} - Y_{estimada_i})^2}{\partial \beta_1} &= 0 \end{aligned} \tag{5.6}$$

Resolviendo el sistema de ecuaciones, se tiene un mínimo para el valor residual cuando se cumple el sistema de Ecuaciones 5.7.

$$\begin{aligned} \beta_1 = \frac{\sum_{k=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{k=1}^{N} (x_i - \bar{x})^2} \\ \beta_0 = \bar{x} - \beta_1 \bar{x} \end{aligned} \tag{5.7}$$

En el caso de la media de los Parámetros K, se desea que la pendiente sea no nula e igual a un valor constante. Esto implicaría que este valor se mantendría constante a lo largo del tiempo y la ROI seguimiento persigue al transmisor correctamente. Sin embargo, dado que tenemos una población relativamente pequeña, la consideración de pendiente nula se debe realizar de forma estadística.

En este caso, el valor de la pendiente β_1 de la Ecuación 5.7, sigue la forma detallada en la Ecuación 5.9.

$$\beta_1 = \frac{Z}{\sqrt{X_k^2 / K}} \tag{5.8}$$

donde:

- Z es una variable aleatoria que sigue una distribución gaussiana.
- X_k^2 es una variable aleatoria que sigue una distribución chi-cuadrado con k órdenes

de libertad. Destacar que $X_k^2 = X_1^2 + X_2^2 + ...$ y para que X_k^2 siga una distribución chi-cuadrado, X_i debe seguir una distribución gaussiana.

Por tanto, β_1 sigue una distribución *Student*. En estas condiciones, para poder determinar con un cierto grado de confianza que esta pendiente es estadísticamente cero se aplica el *test de Student*.

Test de Student

Para realizar el *test de Student* se calcula el parámetro T que viene definido en la Ecuación 5.9.

$$T_0 = \frac{\beta_0 - \beta_{0,0}}{se(\beta_0)}$$
$$se(\beta_0) = \sqrt{\frac{\sum_{k=1}^{N}(y_i - \bar{y})^2}{n-2}\left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{k=1}^{N}(x_i - \bar{x})^2}\right]} \tag{5.9}$$

donde:

- β_1 y β_0 son los parámetros estimados.
- $se(\beta_0)$ es el error estándar del parámetro β_1.
- $\beta_{1,0}$ es la hipótesis nula.

La hipótesis nula es lo contrario de lo que desea comprobar. En este caso, se desea comprobar que existe dependencia entre X e Y. Por tanto, la hipótesis nula es que X e Y sean independientes, esto es, $\beta_1 == 0$. En el caso que la hipótesis nula sea rechazada, nos indica que la hipótesis de contraste, la contraria a la nula, se cumple con un cierto grado de confianza.

Para poder rechazar o no la hipótesis nula, se usa el parámetro T_0. Primero se define el grado de confianza, esto nos implica la certeza de la hipótesis de contraste. Con dicho porcentaje y los grados de libertad, que nos define la forma de la distribución Student, podemos calcular los valores críticos de T. Por tanto, si T_0 queda por encima del valor crítico superior, o por debajo del valor crítico inferior, la hipótesis nula se rechaza y se considera que X e Y son dependientes. En caso de estar entre los valores críticos, la hipótesis nula no se rechaza y se considera que X e Y son independientes.

Por tanto, para rechazar la hipótesis nula, se debe cumplir la Inecuación 5.10.

$$|T_0| > t_{\alpha/2,n-2} \tag{5.10}$$

donde, $t_{\alpha/2,n-2}$ es el valor crítico. Denotar que este parámetro depende tanto de los grados de libertad $n-2$, como del grado de confianza y quedan definidos como aquellos en la distribución donde el área encerrada coincide con la probabilidad de confianza.

5.2. Diseño e Implementación

Esta sección presenta los distintos elementos que componen el sistema de evaluación, así como los parámetros tenidos en cuenta a la hora de diseñar e implementar el entorno.

Inicialmente, dada la naturaleza portable y dinámica de los sistemas *wearables*, es fundamental el estudio del transmisor en movimiento. Es por ello, que se diseña un sistema encargado de trasladar el transmisor a una velocidad constante para tener una mejor comprensión del impacto que supone su desplazamiento. Además, es necesario adaptar el transmisor para su correcta evaluación. Posteriormente, se diseña el sistema de grabación de los vídeos. Y finalmente, se diseña el sistema de procesamiento donde se analizan los vídeos previamente grabados.

Esta sección se divide en cuatro partes, en la que en cada una se discute uno de los bloques de la evaluación:

- **El sistema mecánico**: se presentan el diseño del sistema mecánico cuya función es transladar el transmisor a una velocidad aproximadamente constante.

- **El sistema de transmisión**: se discuten las modificaciones físicas del transmisor.

- **El sistema de grabación**: se detalla el sistema propuesto para la grabación de los vídeos.

- **El sistema de procesamiento**: se presentan los distintos programas para la evaluación del descubrimiento y el seguimiento.

5.2.1. El sistema mecánico

El estudio principal del presente trabajo son los dispositivos *wearables* o portátiles, y como se ha indicado, la propiedad más significativa es la movilidad. Esto implica que se ha de diseñar un sistema que traslade el transmisor, preferiblemente a una velocidad constante.

El sistema mecánico se compone de:

- **Servomotor**: motor responsable de ejercer la fuerza para desplazar el transmisor.
- **Sistema de engranajes**: sistema responsable de convertir el torque del motor en un movimiento rectilíneo uniforme.
- **Generador de Funciones**: sistema responsable de excitar al motor para que se mueva acorde a las especificaciones.

5.2.1.1. Servomotor

Este dispositivo es un motor eléctrico que convierte la energía eléctrica en un movimiento mecánico. Normalmente, el control de todos los motores eléctricos está basado en una PWM (Modulación por Anchura del Pulso). En los motores DC convencionales, según el ciclo de trabajo de la PWM, el motor revoluciona a mayor o menor velocidad, ya que cuanto mayor sea el ciclo, mayor es la cantidad de energía transferida.

En cambio, el uso de la PWM en los servomotores está relacionado con la posición relativa que toma el actuador. Es decir, a cada ciclo de trabajo de la señal PWM le corresponde una posición del servomotor. En nuestro caso, la posición se entiende como un ángulo referido al ciclo de trabajo asignado.

Los servomotores se pueden clasificar en servomotores de salto absoluto o servomotores paso a paso. En los primeros, suponiendo que inicialmente se está a 10° y se quiera pasar a 20°, solo es necesario usar el ciclo de trabajo referido a los 20°. Sin embargo, en el segundo tipo de servomotores, es necesario pasar de 10° a 20° en pasos de, por ejemplo, 1°.

El servomotor elegido en este caso ha sido el paso a paso HS-5685MH por su disponibilidad. Este tipo de servomotor acepta un rango de alimentación de 6.0 V a 7.4 V. Esta tensión está relacionada con el torque, que es el momento que ejerce un motor

sobre su eje. En nuestro caso, se alimenta con una tensión de 7.4 V para obtener un mayor torque. El generador de alimentación usado es el Promax FAC-662B.

5.2.1.2. Sistema de engranajes

En segundo lugar, se encuentra el sistema que transforma el torque del servomotor en un movimiento rectilíneo. Para ello, se ha diseñado y fabricado un sistema de engranajes de dientes rectos. Este sistema se muestra en la Figura 5.3. Se ha usado dos tipos de engranajes, uno consistente en una sección de tipo circular, y otra de tipo lineal. Cabe destacar que la sección circular extiende un ángulo de 90° grados. La velocidad angular generada por el motor se transforma en un movimiento rectilíneo gracias a este sistema de engranajes, la cual toma valor según la Ecuación.

$$V = R * \frac{\theta}{t} \tag{5.11}$$

donde:

- V: velocidad en metros por segundos.
- R: radio del engranaje en metros.
- θ: ángulo que barre el sistema en radianes.
- t: tiempo en el cual se barre dicho ángulo θ.

El radio del engranaje de tipo circular, relacionado con la longitud del engranaje de tipo lineal, está determinado por la fabricación. Mientras que la velocidad angular está determinada por las capacidades del servomotor.

Para el diseño de los engranajes se ha usado el editor profesional de vectores gráficos Inkspace.

Para la fabricación de los engranajes se ha usado una cortadora láser Endless System SLU IL-3000 y placas de metacrilato de 0.4 x 0.5 metros. Por esto, el radio límite obtenido por el plano indicado en la Figura 5.3 ha sido aproximadamente de 0.35 m. Este valor fue determinante a la hora de especificar de la velocidad máxima alcanzable por el sistema.

Finalmente, señalar que el sistema de engranajes también consta de:

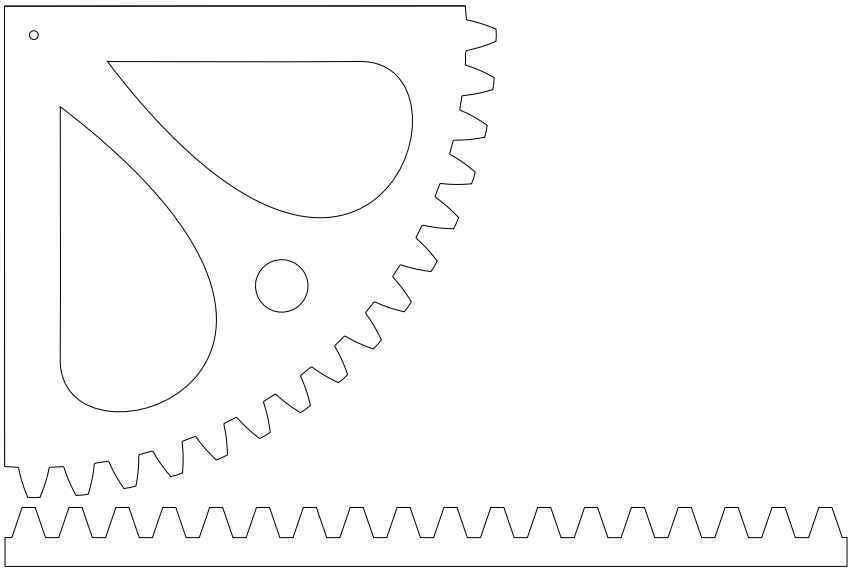

Figura 5.3: Esquemáticos de los engranajes.

1. Una base de soporte de metacrilato. Se usa este material ya que ofrece una menor fricción.

2. Un soporte para el servomotor en madera. Este soporte permite inmovilizar el servomotor haciendo que solamente se mueva el engranaje circular.

3. Una tira de metacrilato que limita el movimiento del engranaje.

El sistema montado completo se observa en la Figura 5.4.

5.2.1.3. Generador de funciones

En tercer lugar, se encuentra el sistema generador de la señal PWM. La señal generada debe tener una tensión de 5 V y un ciclo de trabajo entre 760 y 1710 microsegundos (correspondiente a 0° y 90°). La tensión y la frecuencia de operación está impuesta por el servomotor usado; mientras que los ciclos de trabajo estám impuestos por el sistema de engranajes, ya que este corresponde a la cuarta parte de la circunferencia (90°). Además, dado que el servomotor es paso a paso, el generador ha de barrer los ciclos de trabajo entre 0° y 90°, y viceversa. Para ello, se usa una modulación triangular sobre el ciclo de trabajo. Se ha comprobado empíricamente que el mejor tiempo de barrido, es decir, la frecuencia de modulación de la señal triangular sobre el ciclo de trabajo es de aproximadamente 1 Hz. En caso de especificar una mayor frecuencia, velocidad de barrido, el servomotor no es capaz de realizar el movimiento correspondiente para los

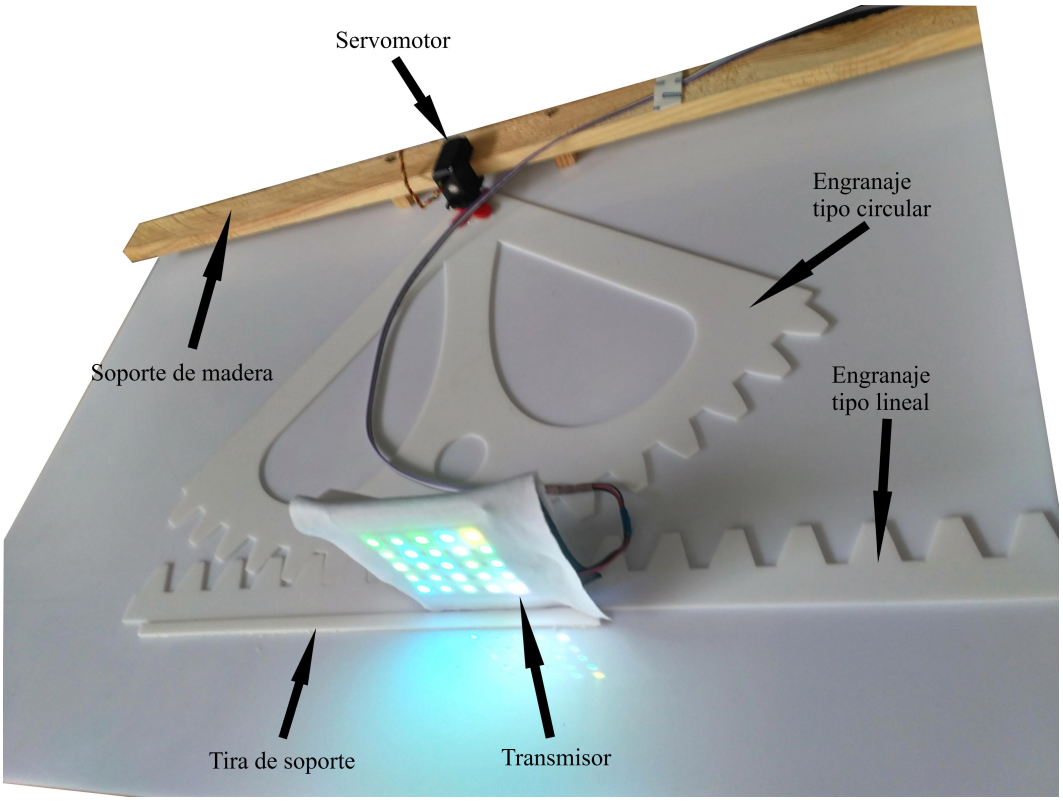

Figura 5.4: Sistema mecánico.

90°. Por tanto, para obtener una mayor velocidad se usa el valor límite de 1 Hz. Esto implica que para un θ de 90° o $\pi/2$ existe un tiempo de 0.5 s (ya que la ida y vuelta serían 1 segundo, correspondiente a 1 Hz). Para la generación de las señales se ha usado el generador de funciones Promax FA-851.

5.2.1.4. Velocidad del sistema

Según las consideraciones de diseño expuestas en los apartados anteriores (un radio de 0.35 metros, un ángulo de 90° y un tiempo de 0.5 s) la velocidad del sistema es aproximadamente 1.1 m/s.

Cabe señalar que, como el sistema de engranajes está limitado en ángulos como se ha indicado, la trayectoria tiene un vaivén de ida y vuelta. Esto produce pequeñas desaceleraciones y aceleraciones . Por tanto, se puede considerar que la velocidad obtenida es en el caso estable donde no existen transiciones. Por otra parte, ya que se usa una modulación triangular, esta transición es mínima por la forma de la señal.

5.2.2. Sistema de transmisión

Teniendo en cuenta el montaje descrito en el sistema mecánico, como se observa en la Figura 5.4, se deben realizar una serie de adaptaciones del sistema de transmisión.

La primera de las modificaciones es relativa al mecanizado del transmisor. Debido al corto espacio disponible para la colocación del transmisor, no es posible situar toda la camisa. Es por ello que se recorta un trozo de tela del mismo tipo que la camisa. Este detalle es importante ya que la radiometría y colometría varía según el tipo de tela, es decir, que afecta al color percibido por el receptor, debido a la absorción de ciertas longitudes de onda y al tipo de mallado de la tela, entre otros factores. La tela se puede observar en la Figura 5.4

La segunda y tercera modificación es relativa al programa del transmisor.

En primer lugar, el programa del transmisor se modifica para que los LEDs transmitan información pseudoaleatoria. Ya que este tipo de datos representan el caso extremo, si se descubre y sigue correctamente el transmisor, se asegura que el sistema es capaz de detectar y seguir al transmisor en cualquier otro caso.

En segundo lugar, también se configuran diferentes tiempos de duración del pulso en base al criterio de tasa de Nyquist, $t_N = 2, 4, 8$. Recordar que la tasa de Nyquist hace referencia al mantenimiento temporal del dato para obtener, al menos, un frame donde el dato es válido sin aplicar técnicas de sincronización.

El programa usado difiere mínimamente del programa expuesto en el Capítulo 3, añadiendo funciones mínimas para el almacenamiento de los datos obtenidos durante las pruebas.

5.2.3. Sistema de grabación

Para la grabación de los vídeos se utiliza una Raspberry Pi con el módulo de Pi Camera V2. Este módulo tiene las siguientes especificaciones hardware:

- 8 Megapíxeles de resolución.
- Sensor de 3280x2464 píxeles.
- Field of View horizontal de 62.2 grados.

- Field of View vertical de 48.8 grados.
- Mecanismo de adquisición basado en rolling shutter.

Además, cuenta con los siguientes modos de operación, que configuran intrínsicamente la resolución, la relación de aspecto y los fps, tal y como se expone en la tabla 5.1.

Tabla 5.1: Modos de operación de la Pi Camera V2.

Modo	Resolución	Relación de aspecto	Rango de FPS permitidos
1	1920x1080	16:9	0.1-30fps
2	3280x2464	4:3	0.1-15fps
3	3280x2464	4:3	0.1-15fps
4	1640x1232	4:3	0.1-40fps
5	1640x922	16:9	0.1-40fps
6	1280x720	16:9	40-90fps
7	640x480	4:3	40-90fps

Existen varios modos de uso de la cámara, mediante la librería 'picamera', de Python, o mediante un programa de consola llamado 'raspivid'. En este caso, se usa este segundo método por su comodidad. Para su uso se implementa un script de bash (extensión '.sh').

El comando 'raspivid'toma como argumentos los siguientes parámetros:

- **w**: ancho de la imagen en píxeles.
- **h**: alto de la imagen en píxeles.
- **o**: nombre del archivo del vídeo. Se escoge el nombre terminado en '.h264', ya que se da este formato al vídeo.
- **t**: duración del vídeo en milisegundos.
- **fps**: fotogramas por segundos del vídeo.
- **ex**: modo de exposición. Se escoge el 'spotlight', debido a la naturaleza del problema. Dicho modo realza o destaca los sistemas de iluminación como el transmisor LED.
- **awb**: modo del balance del blanco. El cual puede ser automático o manual.
- **ss**: tiempo de escaneo en microsegundos.

- **awbg**: valores de las ganancias relativas del rojo y el azul respecto al verde (sólo permitido en modo manual).
- **ag**: ganancia analógica general de la cámara.

Finalmente, indicar que el archivo de vídeo generado tiene formato H264.

El script bash usado tiene la siguiente estructura:

```
#!/bin/bash

w=...             # anchura de la imagen
h=...             # altura de la imagen
time=...          # tiempo de grabación en milisegundos
fps=...           # fotogramas por segundo
ex_md=...         # exposure modes (spotlight)
sh_speed=...      # tiempo de exposición
a_r=...           # ganancia del rojo (balance blanco)
a_b=...           # ganancia del azul (balance blanco)
ag=...            # ganancia analógica de la cámara

vidname=....h264 # nombre del video

raspivid -w \${w} -h \${h} -o \$vidname -t \${time} -fps \${fps} -ex
↪  \${ex_md} -awb off -ss \${sh_speed} -awbg \${a_r},\${a_b} -ag
↪  \${ag}
```

Código 14: Fichero bash para la obtención de los vídeos.

5.2.3.1. Organización de los vídeos

Los vídeos se han realizado a dos distancias, de 1.3 metros y 2 metros con respecto al transmisor. Además, los vídeos se clasifican en tres tipos en base al movimiento que describe el transmisor. En el primer tipo de video, el transmisor se mueve acercándose y alejándose de la cámara (movimiento frontal), esto es, en la dimensión normal a la superficie de la lente de la cámara. En el segundo tipo, el transmisor se mueve paralelamente al plano de la cámara (movimiento lateral), manteniendo constante su distancia, y con una rotación de 0° en el plano de la imagen. Por último, en el tercer vídeo, el transmisor se rota 45 grados en el plano de la imagen (movimiento diagonal),

manteniendo el movimiento descrito en el tipo anterior, a una distancia constante. Por otro lado, para cada tipo de vídeo, se seleccionan dos frecuencias de grabación: 60fps y 90fps.

5.2.4. Sistema de procesamiento

El sistema de procesamiento se divide en tres bloques fundamentales, estos son la prueba de descubrimiento, la prueba de seguimiento y, finalmente, el cálculo de métricas, así que se estudian estos tres bloques de forma separada. Por esto, el programa realizado se divide en tres partes tal y como se muestra en la Figura 5.5.

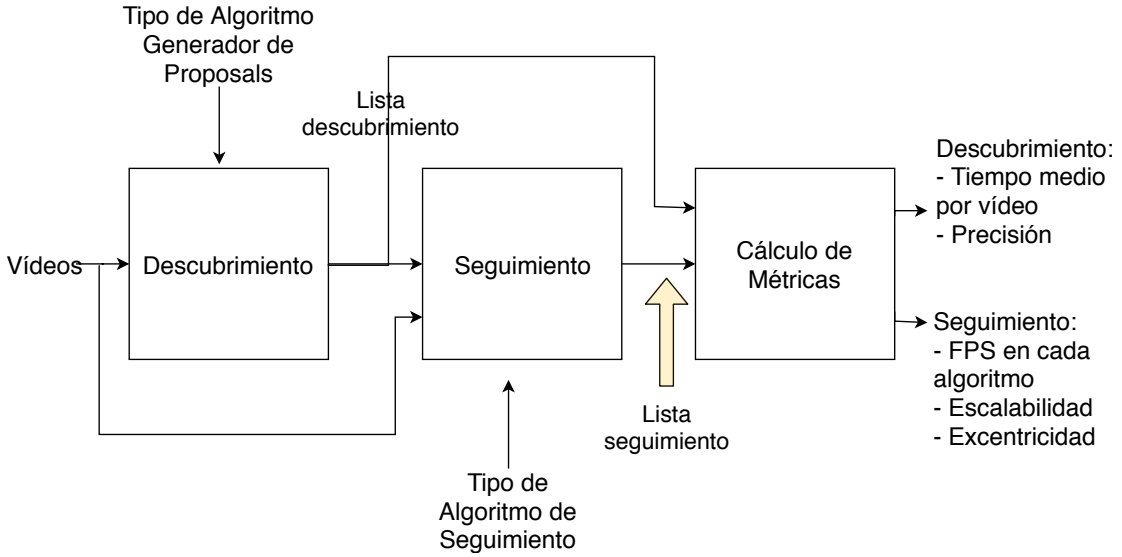

Figura 5.5: Implementación de la evaluación del sistema de procesamiento.

La prueba de descubrimiento toma como entradas el tipo de algoritmo para generación de *proposals*, y una lista de vídeos de muestra; y devuelve una lista con todas las ROIs descubrimiento y el tiempo medio de ejecución.

La etapa de seguimiento toma como entradas el tipo de algoritmo utilizado para el seguimiento, la lista de videos, y la lista de las ROIs encontradas por el descubrimiento. Y devuelve las ROI seguimiento y los FPS en el seguimiento de la fuente para cada vídeo en una lista.

Finalmente, la etapa de cálculo de métricas toma las listas generadas por el descubrimiento y el seguimiento, y se calculan las diferentes métricas de cada prueba. Tiene como salida las métricas de sensibilidad (Equación 5.1) y tiempo medio para el descubrimiento; las métricas de escalabilidad y FPS medio para cada tipo de algoritmo

de seguimiento; y una vez elegido el algoritmo de seguimiento óptimo, se analiza la métrica de excentricidad en cada vídeo.

Cabe mencionar que, para guardar y leer las listas almacenadas en archivos, se hace uso de la librería Pickle de Python. Esta librería tiene dos funciones fundamentales: `pickle.dump` para guardar en disco, y `pickle.load` para leer de disco. Además, en ambos casos se usa la función `open` que recibe por parámetros el nombre del archivo y un indicador de la opción que se va a realizar, en caso de escribir la lista se especifica `'wb'` , en caso de leer se especifica `'rb'` .

5.2.4.1. Descubrimiento

Como se describe en capítulos anteriores, el descubrimiento es la etapa responsable de detectar la fuente luminosa en la imagen, que en este caso es la matriz de 25 LEDs.

El análisis del descubrimiento toma como entradas:

- La lista de vídeos para su análisis.
- El tipo de algoritmo de descubrimiento.

Y devuelve como salidas:

- La lista donde se recogen las ROIs descubrimiento encontradas y los tiempos medios de ejecución de cada frame para todos los vídeos. En el caso de no encontrar la ROI se guarda un objeto `None` . Además, la posición de los datos en la lista indica qué frame se ha analizado, por lo que no es necesario guardar este dato.

En este caso, ya que solo se implementan las etapas de descubrimiento y ajuste de la ROI seleccionada, solo es necesario hacer uso de la clase descubrimiento y la función de ajuste de la roi en la clase principal de AdquisicionSignal. Recordar que la clase AdquisicionSignal es la responsable de coordinar los distintos estados del sistema: el descubrimiento, el ajuste de la ROI seleccionada y el seguimiento.

Por otro lado, esta etapa también necesita guardar los datos de cada frame en una lista que será guardada en disco. Esto permite agilizar el procedimiento de evaluación, ya que permite paralelizar la realización de cada una de las pruebas y el cálculo posterior de métricas.

Clase AdquisicionSignal

La clase descubrimiento y la función ajusteROIseleccionada no han sido modificadas.

Las modificaciones introducidas en la clase AdquisicionSignal se resaltan con marcador verde en la siguiente sección de código.

```
1  class AdquisicionSignal():
2
3      def __init__(self, queue_frame_proc, queue_proc_presenta, ...):
4          self.queue_frame_proc = queue_frame_proc
5          self.queue_proc_presenta = queue_proc_presenta
6
7          self.descubrimiento = Descubrimiento(...)
8
9      def run(self, delta_roi):
10
11         flag_isRunning = True
12
13         lista_descubrimiento = list()
14
15         while flag_isRunning:
16
17             init_time = cv2.getTickCount()
18             flag_isRunning, im = self.queue_frame_proc.get()
19             flag_detectado, roi_descubrimiento =
               ↪  self.descubrimiento.execute(flag_while, im)
20
21             if flag_detectado:
22                 roi_seleccionada = ajusteROIseleccionada(delta,
                   ↪  roi_descubrimiento, im)
23                 self.queue_proc_presenta.put([im,roi_seleccionada])
24                 time = (cv2.getTickCount()-init_time) /
                   ↪  cv2.getTickFrequency()
25                 lista_descubrimiento.append([time, roi_seleccionada])
26
27             else:
28                 self.queue_proc_presenta.put([im,None])
29                 time = (cv2.getTickCount()-init_time) /
                   ↪  cv2.getTickFrequency()
30                 lista_descubrimiento.append([time, None])
31
32         return lista_descubrimiento
```

Código 15: Clase AdquisicionSignal modificada para la prueba de descubrimiento.

Función `init`

En la función init se observa que no se instancia la clase descubrimiento.

Función `run`

En la función run solo se implementa la etapa de descubrimiento. Toma como entrada el incremento de la ROI ajustada para generar la ROI seleccionada. La salida es la lista donde se guardan las ROI seleccionadas encontradas en todos los frames.

En la línea 10 se inicia el flag_isRunning que controla el bucle condicional. En la línea 12 se instancia la lista lista_descubrimiento donde se guardan todas las ROI seleccionadas descubiertas. En la línea 17 se obtiene la imagen de la cola y se actualiza el flag del condicional. En la línea 18 se ejecuta el descubrimiento y se obtiene el flag que indica si se descubrió y la ROI descubrimiento. En la línea 19 se observa si se ha descubierto alguna ROI descubrimiento en el frame. En caso afirmativo, en la línea 21 se ajusta la ROI, en la línea 22 se guarda en la cola Procesamiento-Presentación, y en la línea 24 se guarda la ROI seleccionada descubierta en la lista. En caso de no encontrar ninguna ROI, se guarda un objeto None en la cola en la línea 27, y se guarda otro objeto None en la lista_descubrimiento en la línea 29.

Por otra parte, para la métrica temporal, en la línea 23 se inicializa el timer. Para ello, se hace uso de la función cv2.getTickCount de OpenCV. Posteriormente, en caso de obtener una ROI descubrimiento, en la línea 23, gracias a cv2.getTickCount, el valor inicial, y cv2.getTickFrecuency, se obtienen los segundos que tarda en ejecutarse el descubrimiento. Estos datos son almacenados junto a la ROI (líneas 24 - 29).

Finalmente, cuando flag_while sea False, es decir, cuando no se obtengan más *frames*, se sale del bucle y se retorna la lista_descubrimiento (línea 28).

Programa principal

Por otro lado, también se ha modificado el programa principal. Ya que en este caso el método run de AdquisicionSignal devuelve una lista, el programa principal se encarga de guardar esta lista en disco para su posterior análisis.

La estructura del main es la siguiente:

```
1  if __name__ == '__main__':
2
3      nombredelvideo_o_camara = ...
4
```

```
5       queueEntrada = multiprocessing.Queue()
6       queueSalida = multiprocessing.Queue()
7
8       cl = CamLoop(queueEntrada)
9       ds = AdquisicionSignal(queueEntrada, queueSalida, tipoTracker=4)
10      sl = ShowLoop(queueSalida)
11
12      cam_process = multiprocessing.Process(target=cl.cam_loop,
        ↪  args=(nombredelvideo_o_camara, ))
13      cam_process.start()
14
15      show_process = multiprocessing.Process(target=sl.show_loop)
16      show_process.start()
17
18      lista_descubrimiento = ds.run()
19
20      cam_process.join()
21      show_process.join()
22
23      nombreArchivo = ...
24      with open(nombreArchivo, 'wb') as filehandle:
25          pickle.dump(lista_descubrimiento, filehandle)
```

Código 16: Multiprocesamiento modificado para el descubrimiento.

Los cambios respecto al main principal expuesto en la Implementación residen en las líneas 18, 24 y 25. En la línea 18, al llamar a la función run del objeto de la clase Descubrimiento, se obtiene la lista de las ROI seleccionadas en la variable lista_descubrimiento. En las líneas 24 y 25, la lista se guarda en memoria. En primer lugar, se abre el archivo con un nombre dado y se especifica que se va a realizar una escritura en memoria con 'wb'. En segundo lugar, se guarda en el archivo gracias a la función pickle.dump donde se indica el dato a guardar y el archivo abierto.

5.2.4.2. Seguimiento

El seguimiento se encarga de seguir al transmisor, utilizando uno o más frames de posiciones conocidas.

El análisis del Seguimiento toma como entradas:

- La lista de vídeos.

- La lista devuelta por el descubrimiento.
- El tipo de algoritmo de seguimiento.

Y devuelve como salidas:

- La lista donde se recogen las ROIs seguimiento encontradas en cada frame y los FPS medio en el seguimiento para todos los vídeos.

La ROI seleccionada para la inicialización del seguimiento se obtiene de la lista obtenida del descubrimiento. La ROI coincide en índice con el número del frame analizado.

En este caso, también se modifica la clase AdquisicionSignal utilizando exclusivamente la clase de seguimiento. Por otro lado, la lista devuelta contiene las ROIs seguimiento calculadas para todos los frames posibles. Esta lista se guarda en memoria.

Sin embargo, para poder iniciar el algoritmo de seguimiento, se debe conocer la ROI seleccionada para dicho frame. Por esto, en esta etapa también se lee de memoria la lista obtenida de la etapa del descubrimiento.

Clase AdquisicionSignal

La clase AdquisicionSignal en este caso tiene las siguientes modificaciones:

```
1  class AdquisicionSignal():
2
3      def __init__(self, queue_frame_proc, queue_proc_presenta, ...):
4          self.queue_frame_proc = queue_frame_proc
5          self.queue_proc_presenta = queue_proc_presenta
6          self.seguimiento = Seguimiento(...)
7
8      def run(self, delta, lista_descubrimiento):
9
10         lista_seguimiento = list()
11         flag_isRunning = True
12         estado = 'descubrimiento'
13         indiceFrame = 0
14
15         while flag_isRunning:
16
17             if estado == 'descubrimiento':
```

```
18                  flag_while, im = self.queue_frame_proc.get()
19                  _, roi_seleccionada = lista_descubrimiento[indiceFrame]
20                  indiceFrame += 1
21
22                  if roi_descubrimiento != None:
23                      estado = 'seguimiento'
24                      self.queue_proc_presenta.put([im,roi_seleccionada])
25                      numFrame = 0
26                      lista_seguimiento.append([0,None])
27
28                  else:
29                      self.queue_proc_presenta.put([im,None])
30                      lista_seguimiento.append([0,None])
31
32              else:
33
34                  if ...:
35                      self.seguimiento = Seguimiento(self.tipoTracker)
36                      flag_tracker = ⌋
                        ↪ self.seguimiento.init_tracker(im,roi_seleccionada)
37                      if flag_tracker:
38                          init_time = cv2.getTickCount()
39                      else:
40                          init_time = cv2.getTickCount()
41                          numFrames = X - 1
42
43                  flag_while, im = self.queue_frame_proc.get()
44                  indiceFrame += 1
45                  numFrames += 1
46
47                  if numFrames = X:
48                      self.queue_proc_pres.put([im,None])
49                      estado = 'descubrimiento'
50                      del tracker
51                      time = (init_time -
                        ↪ cv2.getTickCount())/cv2.getTickFrecuency()
52                      fps = numFrames/time
53                      lista_seguimiento.append([fps,None])
54
55                  else:
56
57                      flag_encontrado, roi_seguimiento =
                        ↪ self.seguimiento.update(flag_while, im)
```

```
58
59                      if flag_encontrado:
60
61                          if len(contours) == 25:
62                              self.queue_proc_pres.put([im,roi_seguimien
                                ↪ to])
63                              lista_seguimiento.append([im,roi_seguimien
                                ↪ to])
64
65                          else:
66                              self.queue_proc_pres.put([im,None])
67                              estado = 'descubrimiento'
68                              del tracker
69                              time = (init_time - cv2.getTickCount())/cv
                                ↪ 2.getTickFrecuency()
70                              fps = numFrames/time
71                              lista_seguimiento.append([fps,None])
72
73                      else:
74                          self.queue_proc_pres.put([im,None])
75                          estado = 'descubrimiento'
76                          del self.seguimiento
77                          time = (init_time -
                            ↪ cv2.getTickCount())/cv2.getTickFrecuency()
78                          fps = numFrames/time
79                          lista_seguimiento.append([fps,None])
80
81          return lista_seguimiento
```

Código 17: Clase procesamiento modificada para la prueba de seguimiento.

En este apartado, solo se explicará las modificaciones resaltadas en la clase AdquisicionSignal.

En la línea 6 se observa que solo se instancia el seguimiento. En la línea 8 se observa que la función run obtiene por parámetros la lista de las ROIs seleccionadas necesarias para iniciar el seguimiento. Para poder obtener la ROI adecuada de la lista, es necesario un índice que indique el número de frame estudiado. Este índice coindice con la posición de la ROI seleccionada en la lista_descubrimiento (línea 13). Se puede destacar que, cada vez que se obtiene un frame de la cola Frame-Procesamiento, se incrementa en

una unidad (líneas 20 y 44). En este caso, la comprobación de la detección no se realiza con el flag devuelto por el objeto de la clase Descubrimiento, sino que se comprueba si la ROI obtenida de la lista es igual a None. En caso contrario se obtiene la ROI seleccionada.

En la lista_seguimiento se guardan dos parámetros: los fps del seguimiento y ROI seguimiento.

En la línea 28 se observa que se implementa un contador de tiempo que hace uso de la función cv2.getTickCount. Esta función cuenta el número de ciclos de reloj. Por esto, si se restan dos valores de cv2.getTickCount y se divide entre cv2.getTickFRecuency, la frecuencia, se obtiene el tiempo en segundos entre las dos medidas de ciclos de reloj. Estos valores de tiempo son convertidos en FPS dividiendo el número de frames (variable numFrames) entre el tiempo transcurrido. Este valor de FPS es almacenado junto a la ROI seguimiento.

Sin embargo, mientras se encuentre en la etapa de seguimiento, no se guarda constantenemente el valor de FPS. Esta escritura se realiza cuando el seguimiento finaliza, esto es, se pierde al transmisor y se pasa al descubrimiento, o cuando termina el vídeo (líneas 63-79).

Finalmente, en la línea 70, la función run retorna la lista seguimiento generada.

Programa principal

Por otro lado, se ha modificado el programa principal. En este caso también se guarda la lista de seguimiento en disco para que en la etapa de cálculo de métricas se computen las métricas seleccionadas.

La estructura del programa se muestra en la siguiente sección de código:

```
1  if __name__ == '__main__':
2
3      nombredelvideo_o_camara = ...
4
5      queueEntrada = multiprocessing.Queue()
6      queueSalida = multiprocessing.Queue()
7
8      cl = CamLoop(queueEntrada)
9      ds = AdquisicionSignal(queueEntrada, queueSalida, tipoTracker=4)
10     sl = ShowLoop(queueSalida)
11
```

```
12      cam_process = multiprocessing.Process(target=cl.cam_loop,
        ↪  args=(nombredelvideo_o_camara, ))
13      cam_process.start()
14
15      show_process = multiprocessing.Process(target=sl.show_loop)
16      show_process.start()
17
18      lista_seguimiento = ds.run()
19
20      cam_process.join()
21      show_process.join()
22
23      nombreArchivo = ...
24      with open(nombreArchivo, 'wb') as filehandle:
25          pickle.dump(lista_seguimiento, filehandle)
```

Código 18: Multiprocesamiento modificado para la prueba de seguimiento.

La única diferencia con el programa principal de la etapa de descubrimiento es que se obtiene y guarda en disco la lista seguimiento.

5.2.4.3. Cálculo de Métricas

En esta etapa se desarrolla una serie de *scripts* en `Python` para el cómputo de las métricas descritas al inicio de esta sección. En lo que resta de sección se expone cuál es el procedimiento para la obtención de cada una de las métricas.

5.2.4.3.1. Métricas de la prueba de descubrimiento La obtención de las métricas del descubrimiento solo necesita como entrada la lista devuelta por el descubrimiento, de la que se obtiene los siguientes parámetros:

- **Sensibilidad del sistema**: probabilidad de encontrar correctamente el transmisor LED. Se usa la métrica de sensibilidad expresada anteriormente.
- **Tiempo medio de ejecución**: tiempo medio de ejecución del descubrimiento para todos los fotogramas de los distintos vídeos.

5.2.4.3.2. Métricas de la prueba de seguimiento En primer lugar, en el descubrimiento, después que el descubrimiento obtiene la lista de ROIs encontradas, se analiza la calidad de los distintos algoritmos de seguimiento.

- **FPS medio**: Fotogramas Por Segundos medio del programa para cada algoritmo de seguimiento.
- **Escalabilidad**: parámetro binario que nos indica si el algoritmo de seguimiento es escalable. Es decir, si es capaz de aumentar el área cuando el transmisor se acerca, y de disminuir el área cuando se aleja. Se expresa en Sí o No.

En segundo lugar, una vez elegido el algoritmo de seguimiento más eficiente, se usan las métricas relativa a los parámetros K para estudiar el comportamiento del seguimiento. Estas son:

- **Media de los parámetros K**: nos indica el parecido entre áreas: ROI descubrimiento y ROI seguimiento.
- **Desviación estándar de los parámetros K**: nos indica la centralidad de la ROI seguimiento respecto a la ROI seguimiento.

Recordar que se desea una media no nula, constante e igual al parámetro indicado, y una varianza nula.

Posteriormente, se calcula la Regresión Lineal de la media y la desviación estándar de los Parámetros K respecto a la cantidad total de frames del vídeo y se calcula el parámetro T_0 del Test de Student. Finalmente, para una probabilidad de confianza del 95 % y junto con los grados de libertad, se obtienen los valores críticos $T_{\pm\alpha/2,n-2}$, y se comprueba si el parámetro T_0 calculado rechaza o no la hipótesis nula de independencia.

Cabe mencionar que, además de calcular la Regresión Lineal, también calculamos la desviación estándar de los valores residuales. Este parámetro nos indica cuál es el ruido de los valores reales sobre el valor estimado. En caso de tener un valor alto, existe mayor probabilidad de que las muestras reales sean valores negativos. Esto conlleva a un tiempo de vuelta al descubrimiento menor. La desviación estándar de los valores residuales se calcula tanto para la media como para la desviación estándar de los parámetros K.

5.3. Parámetros del Test

En este apartado se resumen los valores de los parámetros utilizados en la configuración del entorno de evaluación.

5.3.1. Sistema de transmisión

Los parámetros del sistema de transmisión son:

- **Tipo de Movimiento**: los movimientos elegidos son frontal, lateral y diagonal.
- **Tasa de Nyquist**: se eligen los valores de 2, 4, 8 respecto al tiempo de frame $1/f_N$.

5.3.2. Sistema de grabación

Los parámetros del sistema de grabación son:

- **Distancia**: se eligen los valores de 1.3 y 2 metros. Se refiere a la distancia entre el transmisor y el receptor.
- **FPS**: fotogramas por segundo. Se elige 60 y 90 FPS.
- **Duración del vídeo**: se elige un tiempo de duración de vídeo extenso para asegurar una alta fiabilidad en la métrica de sensibilidad del descubrimiento.
- **Modo exposición**: el modo spotlight realza o destaca la iluminación. Modo usado por la naturaleza del transmisor.
- **Tiempo de exposición**: es el tiempo durante el cual se realiza el escaneo de la escena. Si se especifica un tiempo de exposición alto, la imagen se integra durante más tiempo, por lo que se pueden observar los objetos menos luminosos, en cambio los objetos más luminosos saturan.
- **Modo balance de blancos**: el modo del balance de blancos debe ser manual, y las ganancias deben ser fijadas antes de la captura. En caso contrario, al tener una imagen oscura, debido al tiempo de obturación bajo, y una fuente luminosa muy variable en color, la cámara intentará realizar el balance en cada frame, lo que provoca que no se detecte correctamente el valor de los canales RGB de los LEDs que componen la fuente.
- **Ganancias del balance de blancos**: se usan dos valores, el balance del rojo respecto al verde, y el azul respecto al verde.

5.3.3. Sistema de procesamiento

El único parámetro del sistema de procesamiento referido al sistema de adquisición de la señal es:

- **Delta**: incremento en píxeles que sufre la ROI descubrimiento ajustada para generar la ROI seleccionada.

5.3.4. Tabla de Métricas

La tabla 5.2 resume los parámetros seleccionados en la configuración.

Tabla 5.2: Tabla de métricas.

	Descripción	Valores
Tipos Movimiento	Tipo de movimiento que realiza el transmisor.	Mov. Normal Cámara, Mov. 0°, Mov. 45°
Distancia	Distancia del enlace en metros.	1.3, 2
FPS	Fotogramas Por Segundos.	60, 90
Delta	Incremento en píxeles de la ROI ajustada.	5, 10, 15
Tasa de Nyquist	Tasa de Nyquist en el receptor referido a los FPS.	f_N, $f_N/2$, $f_N/4$
Ancho (w)	Ancho de la imagen en píxeles.	640
Alto (h)	Alto de la imagen en píxeles.	480
Nombre Archivo (o)	Nombre del Archivo de vídeo en formato H264.	nombreArchivo.h264
Tiempo Grabación (t)	Tiempo de duración del vídeo en milisegundos.	10000
Modo Exposición (ex)	Modo de exposición de grabación.	spotlight
Modo Balance Blanco (awb)	Tipo de modo automático de balance del blanco.	off
Tiempo de exposición (ss)	Tiempo de exposición de la cámara.	85
Valores Balance Blanco (awbg)	Valores introducidos manualmente del balance del rojo y el azul respecto al verde.	3,2
Ganancia Analógica (ag)	Ganancia analógica de la cámara.	3

Capítulo 6

Discusión de los resultados

En este capítulo se presentan y discuten los resultados obtenidos utilizando el sistema de evaluación descrito en el capítulo 5. La discusión de los resultados se divide en dos secciones separadas, una para cada prueba realizada: el descubrimiento y el seguimiento. Los resultados fueron obtenidos utilizando un ordenador con las siguientes especificaciones técnicas:

- **Procesador**: Intel Core i7-8550U. 1.8 GHz con TurboBoost a 4.0 GHz.
- **Memoria RAM**: 8 GB DDR4.
- **Tarjeta gráfica**: NVIDIA GeForce MX139 con 2 Gb de VRAM.
- **Tipo de disco duro**: HDD.
- **Sistema operativo**: Windows 10.

6.1. Descubrimiento

Tal y como se describió en el capítulo 5, las métricas analizadas para la fase de descubrimiento son: la sensibilidad y el tiempo medio de ejecución.

La organización de los resultados sigue un esquema jerárquico en forma de árbol. En primer lugar, se muestran las diferentes gráficas obtenidas para cada tipo de movimiento: lateral, frontal o diagonal. En estas gráficas se presentan los resultados obtenidos para las diferentes distancias: 2 y 1.3 metros, para las dos frecuencias de escaneo de la

cámaraFPS (60 y 90 fps) y para las distintas tasas de Nyquist asociadas ($1/FPS$: 2, 4 y 8) con colores turquesa, azul estándar y azul marino para cada caso respectivamente.

En el caso particular del movimiento frontal, solo se realizaron tres grabaciones, utilizando únicamente la frecuencia de captura de 90 FPS y una frecuencia de transmisión que dobla la captura. Los parámetros para cada vídeo son:

- **Vídeo 1**: velocidad de 0.5 m/s aproximadamente, a una distancia de 1.3 metros.
- **Vídeo 2**: velocidad de 1.1 m/s aproximadamente, a una distancia de 1.3 metros.
- **Vídeo 3**: velocidad de 1.1 m/s aproximadamente, a una distancia de 2 metros.

6.1.1. Sensibilidad

El estudio de la sensibilidad (Recall) se presenta en el Capítulo 5.

Los resultados obtenidos para el movimiento lateral, frontal, y diagonal se muestran en las Figuras 6.1, 6.2, 6.3 respectivamente.

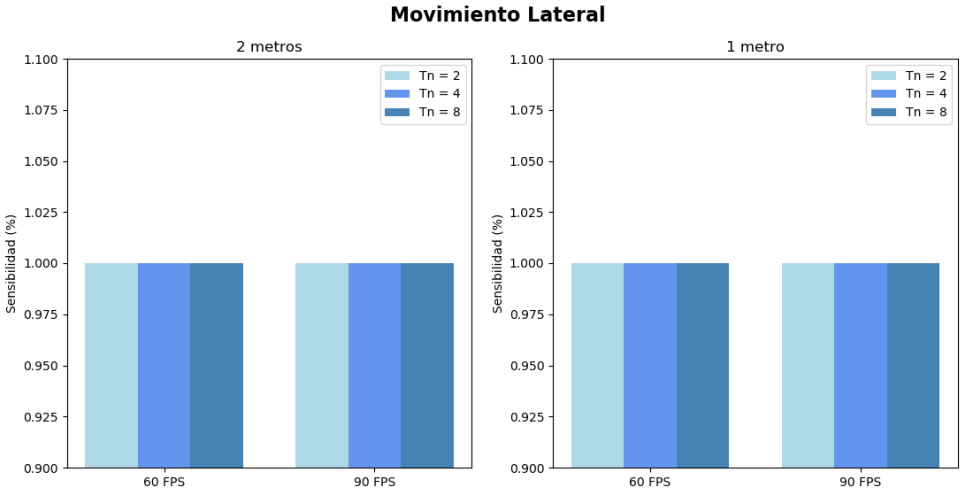

Figura 6.1: Sensibilidad para el movimiento lateral.

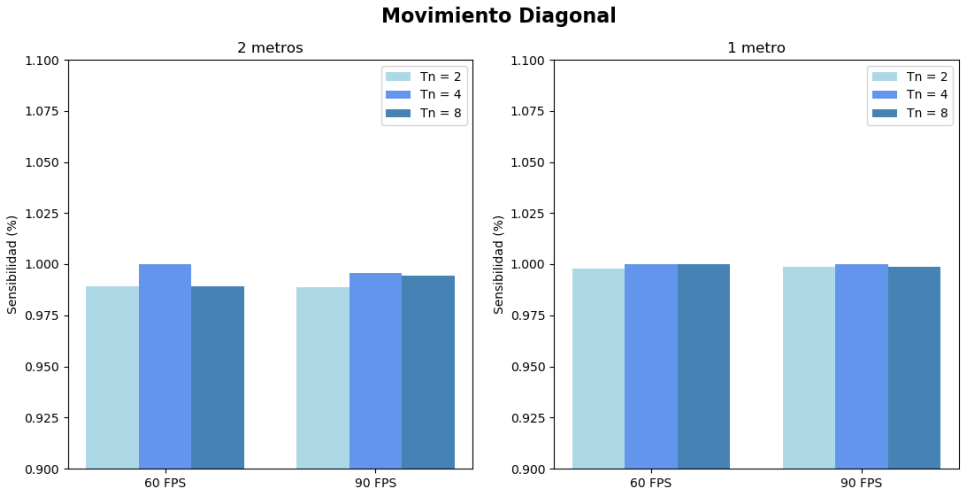

Figura 6.2: Sensibilidad para el movimiento diagonal.

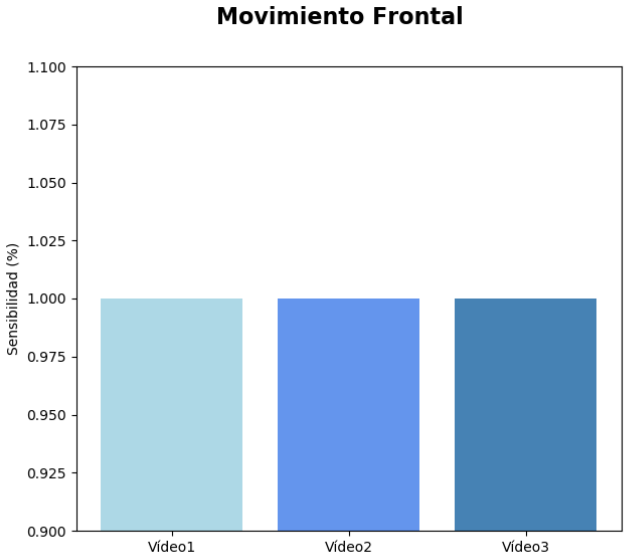

Figura 6.3: Sensibilidad para el movimiento frontal.

De estas figuras, la conclusión principal es que los distintos parámetros de configuración utilizados: tipo de movimiento, distancia, FPS y tasa de Nyquist, no afectan de forma apreciable a la sensibilidad del sistema, siendo el valor próximo o igual al 100 % en la mayoría de los casos.

Se puede concluir que, siempre y cuando exista una visión directa del transmisor, y en las condiciones de interior en las que se realizó el experimento, el descubrimiento detectará perfectamente al transmisor. Esto es así, puesto que se ha aprovechado la naturaleza del objeto como fuente luminosa directa, lo que permite reducir la sensibilidad de la cámara y reducir los fenómenos de interferencia que dificultan la detección

del transmisor.

Cabe destacar que, tal y como se ha llevado a cabo la clasificación de la fuente transmisora mediante la comprobación del contenido de los 25 contornos en la fase de descubrimiento, se puede establecer que la presencia de otras fuentes luminosas en la imagen no afectará de forma apreciable a la precisión del sistema a la hora de detectar fuentes legítimas. Sin embargo, la medida de este valor de precisión no ha sido realizada en este trabajo.

6.1.2. Tiempo medio de ejecución

El tiempo medio de ejecución es el parámetro usado para estudiar el impacto computacional y temporal del descubrimiento. El estudio que se presenta en este apartado nos permite establecer el límite superior de la frecuencia de captura (FPS) de la cámara que esta puede alcanzar si el receptor permaneciese en un estado de descubrimiento continuo de la fuente.

En las Figuras 6.4, 6.5, 6.6 se muestran los resultados obtenidos para el tiempo medio de ejecución para los casos de movimiento lateral, diagonal y frontal.

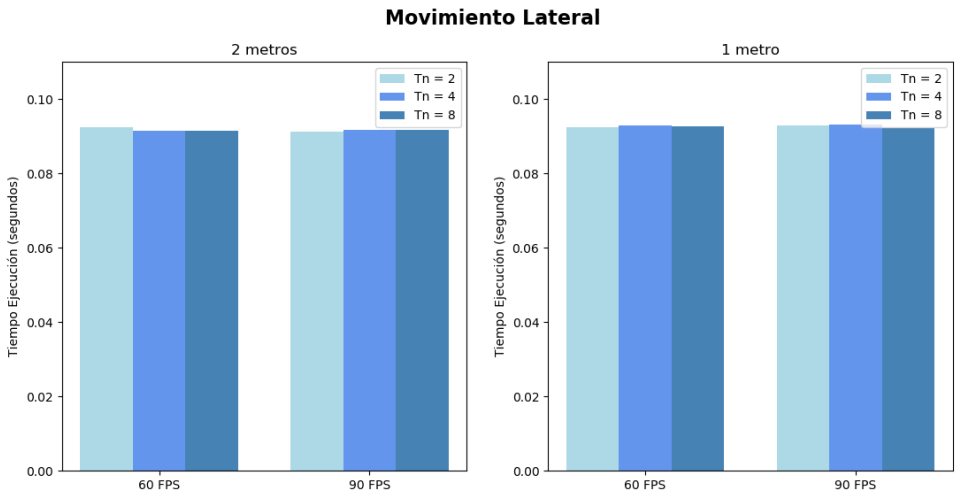

Figura 6.4: Tiempo de ejecución para el movimiento lateral.

Como se puede resaltar de las figuras anteriores, existe un comportamiento idéntico a la sensibilidad en el sentido de que el tiempo de ejecución permanece constante con independencia de la frecuencia de captura de la cámara, lo cual está relacionado con el hecho de que la imagen siempre se recibe con la misma resolución. El valor obtenido para el tiempo de ejecución es de aproximadamente 96 milisegundos, considerablemente

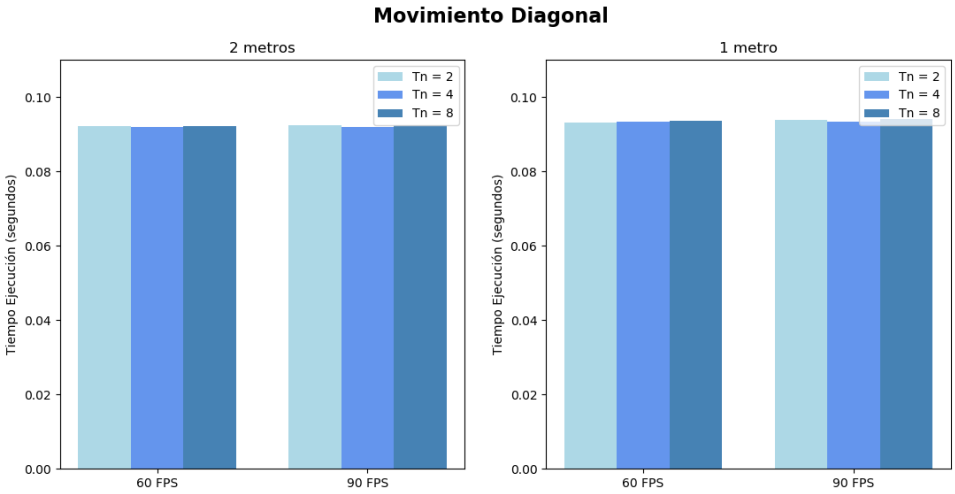

Figura 6.5: Tiempo de ejecución para el movimiento diagonal.

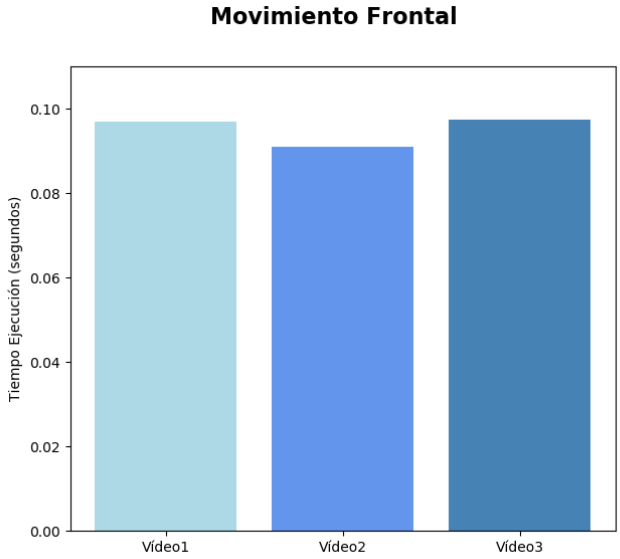

Figura 6.6: Tiempo de ejecución para el movimiento frontal.

mayor al tiempo de adquisición de un frame. En términos de frecuencia, esto implicaría que el sistema sólo podría procesar imágenes con una tasa de captura de $FPS = 1/t_{ejecucion}$. Se puede concluir, por tanto, que, siempre y cuando exista una visión directa del transmisor, en las condiciones del experimento el descubrimiento continuo se puede realizar con tasas de captura entre 10 y 11 FPS, lo cual afecta considerablemente a la tasa máxima de transferencia de datos.

El uso de *Edge Boxes* frente al uso de *Selective Search* ha permitido reducir considerablemente el tiempo medio de ejecución, sin afectar considerablemente a la sensibilidad ya que este se adapta muy bien a la naturaleza de la imagen.

6.2. Seguimiento

Tal y como se comenta en los capítulos 4 y 5, y de las conclusiones obtenidas en la sección anterior se hace necesario implementar un algoritmo que permita el seguimiento de la fuente. Este algoritmo debe tener en cuenta el cambio de escala de la fuente, esto es que la ROI seguimiento debe variar su área acordemente a la proyección de la fuente y debe aceptar velocidades de movimiento relativamente moderadas, dado que se trata de una fuente portátil. Además, se busca aquel algoritmo cuya rapidez de ejecución permita procesar capturas con mayores FPS. Para ello se definen los FPS medio, que indican la máxima tasa de captura que puede soportar el receptor si este sólo se encargase de seguir a la fuente para la localizarla en el frame.

En base a estas consideraciones en la tabla 6.1 se recogen los resultados obtenidos para los FPS medio, junto con un indicador de si el algoritmo soporta o no la escalabilidad de la fuente rastreada.

Tabla 6.1: Tabla del análisis para cada algoritmo de seguimiento.

Algoritmo	FPS medio	Escalabilidad
Boosting	30	No
MIL	17	No
KCF	83	No
TLD	28	Sí
MedianFlow	221	Sí
Mosse	761	No
CSRT	28	Sí

De esta tabla se puede resumir que los algoritmos de seguimiento que presentan escalabilidad son: el TLD, el Median Flow y el CSRT. Entre ellos, el que presenta un mayor valor de FPS medio es el Median Flow.

Por tanto, se puede concluir a-priori que el algoritmo de seguimiento óptimo, bajo ambos criterios de selección es el Median Flow.

En base a estos resultados el estudio de los parámetros K se ajustará exclusivamente a este algoritmo de seguimiento. Este estudio se divide en el análisis temporal de la media y la desviación estándar de dichos parámetros.

Además, es importante señalar que los FPS obtenidos para este algoritmo de seguimiento es considerablemente superior a los obtenidos para el algoritmo de descubrimiento (221 FPS frente a 10-11 FPS).

Esto permite que el sistema reduzca su carga computacional y disponga de recursos para adaptarse a capturas más rápidas, con lo que se aumenta considerablemente la tasa binaria de transmisión tal y como se expresa en la Ecuación 3.4. Además, también facilita el despliegue del sistema receptor en un mayor número de dispositivos que presenten una capacidad computacional más reducida, tales como dispositivos móviles.

6.2.1. Parámetros K

Tal y como se detalla en la introducción del presente capítulo, el estudio de los parámetros K se divide en dos partes: el análisis temporal de la media (escala) y el análisis de la desviación estándar (centralidad). En ambos casos se estudia la pendiente de la regresión lineal obtenida para la evolución temporal, esto es para todos los frames del vídeo, y la desviación estándar del valor residual de esta regresión. La primera indica la tendencia temporal del algoritmo de seguimiento a perder las condiciones óptimas de seguimiento (tanto para la escala como para la centralidad), y la segunda la desviación de los valores reales sobre la tendencia, lo cual indica que durante mayor es la desviación, mayor es la diferencia de las ROIs seguimiento entre frames.

6.2.1.1. Movimiento lateral

En las Figuras 6.7, 6.8 y 6.9 se muestran algunos ejemplos de la tendencia del algoritmo de seguimiento en la localización de la fuente transmisora para el movimiento lateral, en este orden: los casos de pendientes negativa, estable y positiva.

El caso de pendiente negativa indica que la región tiende a estrecharse cada vez más a la fuente. En este caso se observa una desviación de la tendencia mayor. Esto indica, que entre frames cercanos la localización de la fuente difiere considerablemente. Esto puede llevar a casos en los que los valores de K medio resulten negativos, lo cual significa que la fuente ha sido localizada parcialmente, y es necesario volver a realizar el descubrimiento.

En el caso de pendiente estable, significa que el algoritmo de seguimiento es capaz de seguir durante un tiempo mayor a la fuente sin la necesidad de realizar descubrimientos intermedios. Este es el caso más favorable, siempre y cuando la varianza de los valores residuales permanezca pequeña, esto es, que no existan variaciones muy bruscas entre los resultados para cada ROI.

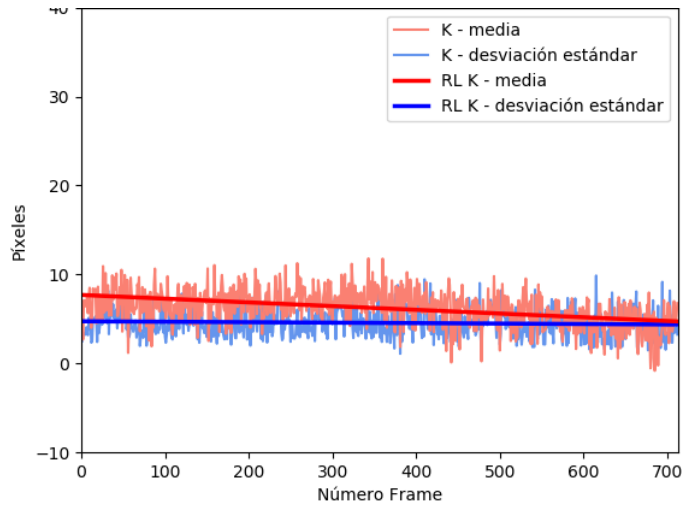

Figura 6.7: Ejemplo de pendiente negativa en el movimiento lateral.

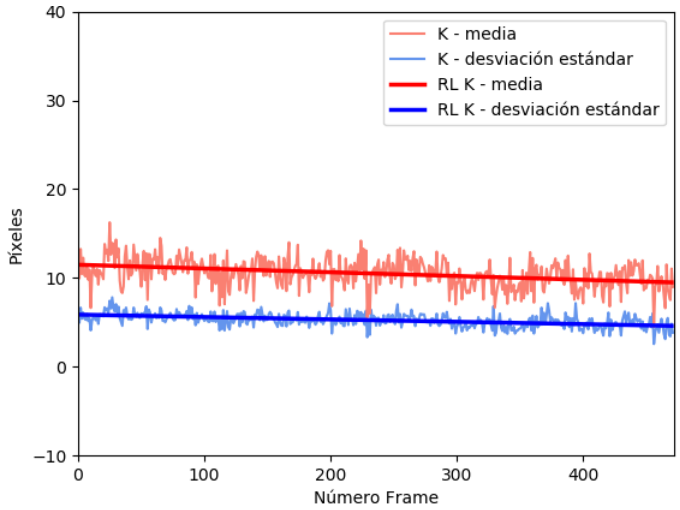

Figura 6.8: Ejemplo de pendiente estable en el movimiento lateral.

En el caso de pendiente positiva, la ROI seguimiento tenderá cada vez a hacerse mayor aun cuando el objeto no ha variado su tamaño, lo cual tiene implicaciones negativas en el sentido de que se conoce de forma menos precisa la ubicación de la fuente. Además, esto implica que el algoritmo es más susceptible a la interferencia de otras fuentes luminosas, esto se debe a que la ROI más grande con el tiempo. Cabe destacar que, en el caso de la camisa, ya que el transmisor se posiciona sobre una persona y dada la reducida sensibilidad de la cámara, el cuerpo humano se comporta como un marco oscuro que permite un margen de incremento de la ROI seguimiento sin que sufra interferencias de otras fuentes externas.

Las Figuras 6.10, 6.11, 6.12 y 6.13 muestran el resumen de estos parámetros temporales para todos los vídeos realizados, siguiendo la estructura expuesta al inicio de

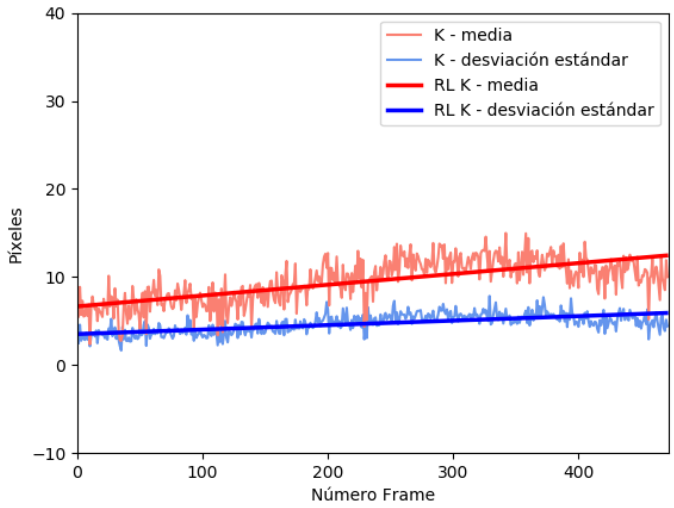

Figura 6.9: Ejemplo de pendiente positiva en el movimiento lateral.

sección.

Sobre la tendencia temporal de la escala en el movimiento lateral se puede observar que, en la mayoría de los casos, los valores tienden a cero, lo que significa que el sistema se comporta de forma relativamente estable.

De igual forma, la desviación de la tendencia de la escala, sigue el mismo comportamiento independientemente de la delta elegida (casi todos los valores son próximos a 2 píxeles). Esto indica que los parámetros de distancia, delta y FPS no afectan en el rendimiento del sistema implementado para el movimiento lateral. También implica que una mayor delta seleccionada protege mejor al seguimiento a la hora de perder la fuente, ya que la desviación de la tendencia es relativamente menor y, por tanto, también lo es la probabilidad de pérdida.

Sobre la tendencia temporal de la centralidad en el movimiento lateral, también se observa que existe independencia entre la configuración del sistema y el rendimiento.

Además, cabe señalar que los valores de la tendencia temporal en la centralidad son menores que en la escala. Esto implica que el sistema para el movimiento lateral es más vulnerable a los efectos debidos a la escala, que los debidos a la pérdida centralidad.

Finalmente, se puede observar que, si la tendencia temporal de la escala es positiva, la tendencia de la centralidad también es positiva, y viceversa. El sentido físico es que si la ROI seguimiento disminuye en área, esta implícitamente queda más centrada. En caso contrario, si la ROI seguimiento tiende a aumentar en área, esta se vuelve más excéntrica respecto a la ROI descubrimiento. Esto es debido a que el algoritmo de

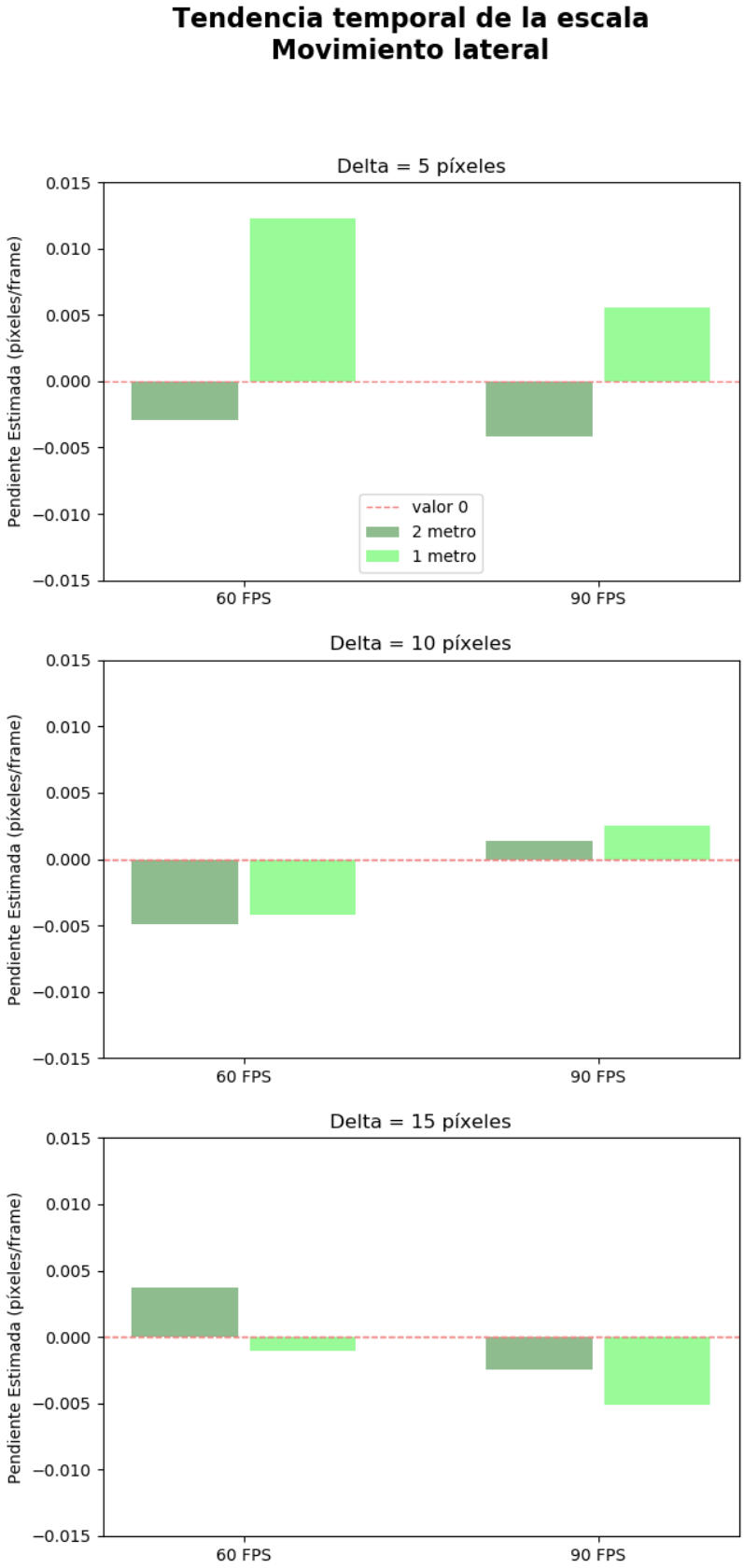

Figura 6.10: Tendencia temporal de la escala para el movimiento lateral.

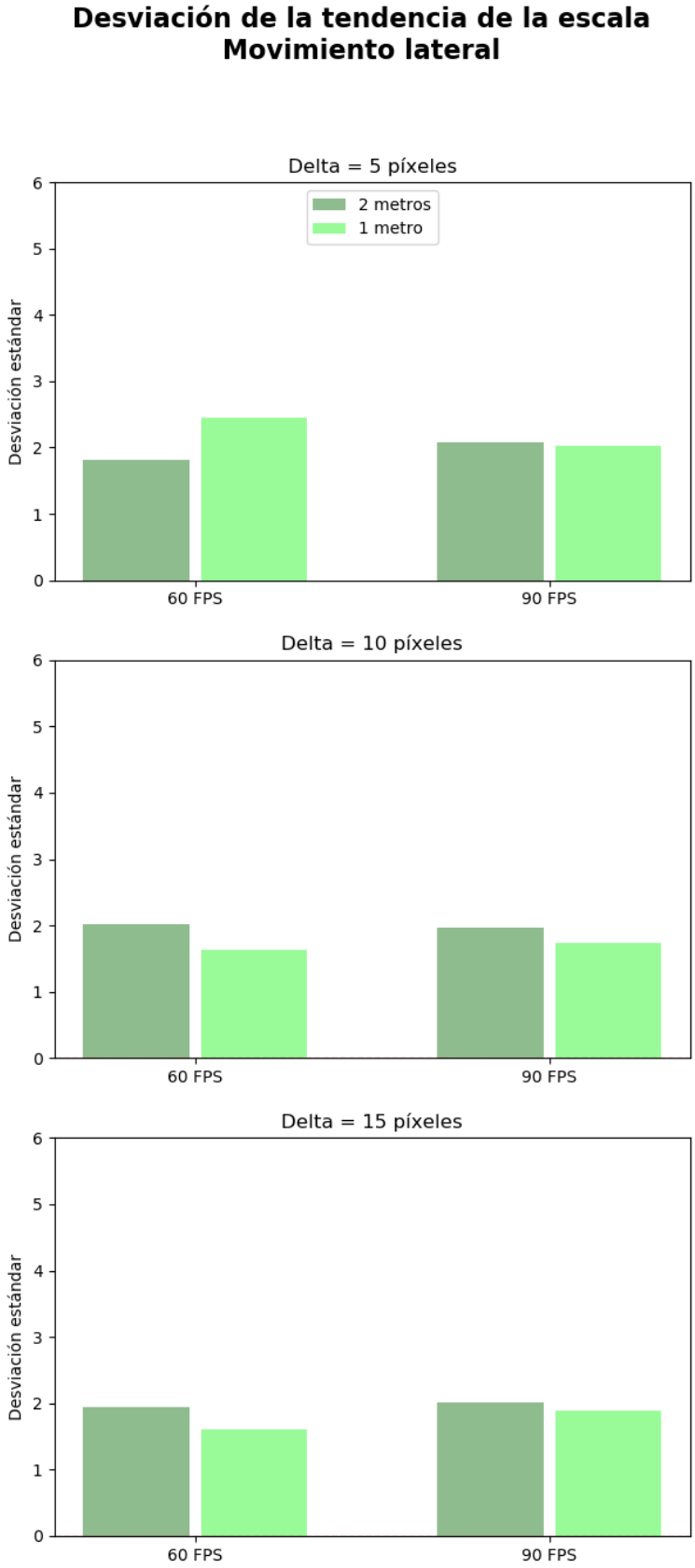

Figura 6.11: Desviación de la tendencia de la escala para el movimiento lateral.

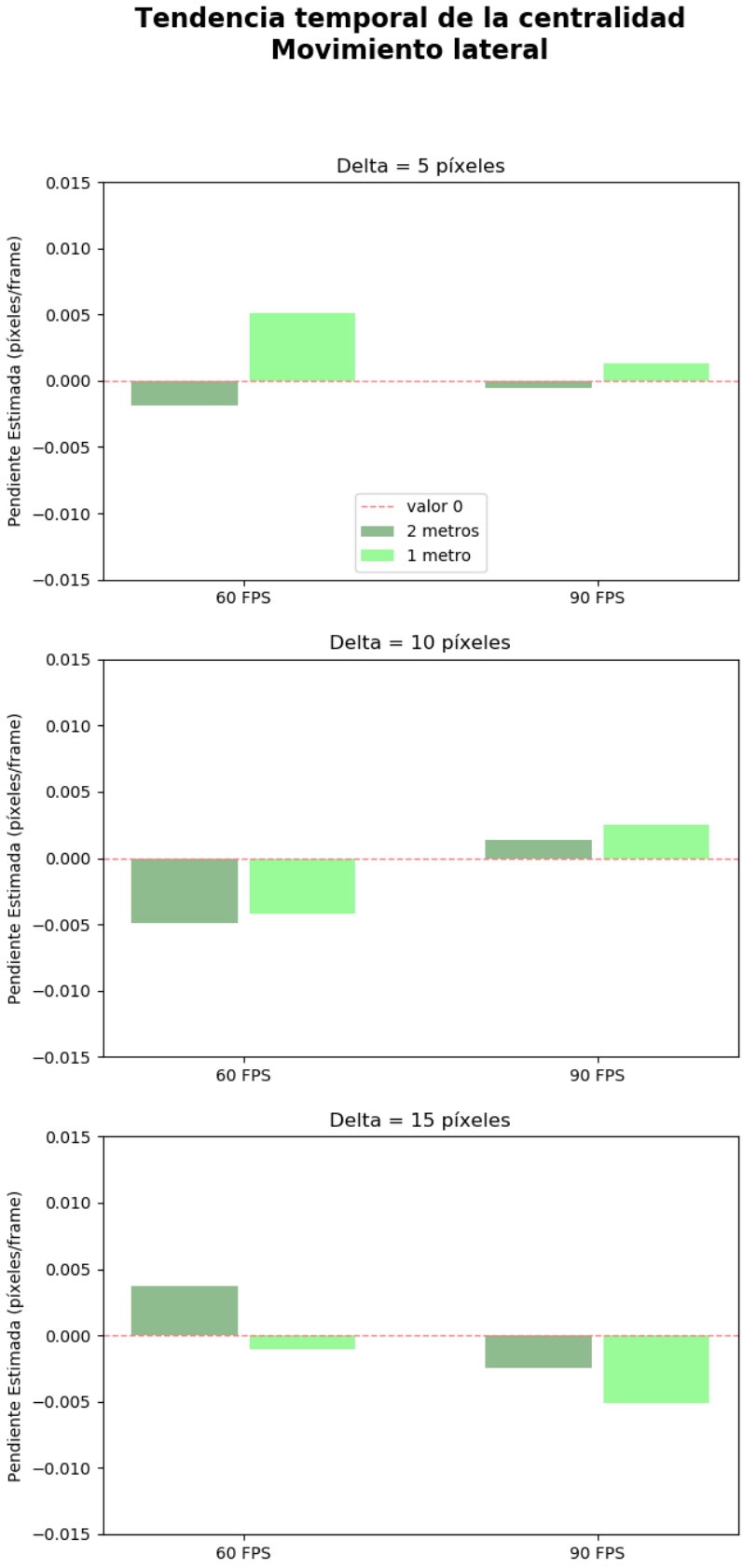

Figura 6.12: Tendencia temporal de la centralidad para el movimiento lateral.

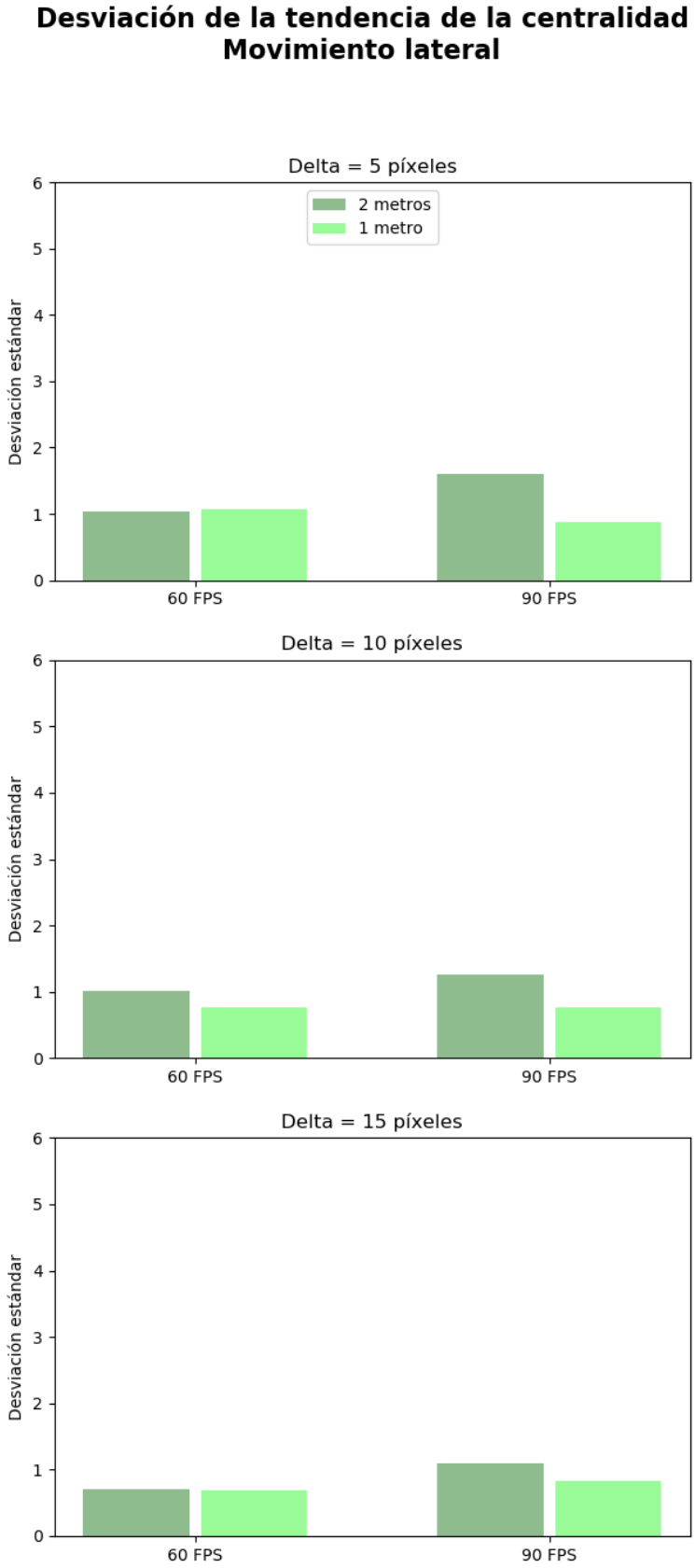

Figura 6.13: Desviación de la tendencia de la centralidad para el movimiento lateral.

seguimiento es menos eficaz a la hora de detectar una fuente transmisora que ocupa menos área que la ROI seguimiento.

6.2.1.2. Movimiento diagonal

En las Figuras 6.14, 6.15 y 6.16, de igual forma que para el movimiento lateral, se muestran algunos ejemplos de la tendencia del algoritmo de seguimiento para el movimiento diagonal.

En el caso de pendiente negativa, se puede observar que las desviaciones de las tendencias de la escala y la centralidad son mucho mayores que para el movimiento lateral. Esto implica que para los mismos parámetros del sistema, este movimiento presenta muchos casos de valores que implican pérdida del transmisor. Los valores negativos implican la pérdida parcial de la fuente y los valores altos positivos implica menor precisión en la localización de la fuente. Por tanto, en estos casos es necesario la ejecución del descubrimiento nuevamente, lo que provoca que los FPS totales del sistema se reduzcan drásticamente. Además, comparado con las otras pendientes, se observa que para pendientes negativas existen un mayor número de casos extremos.

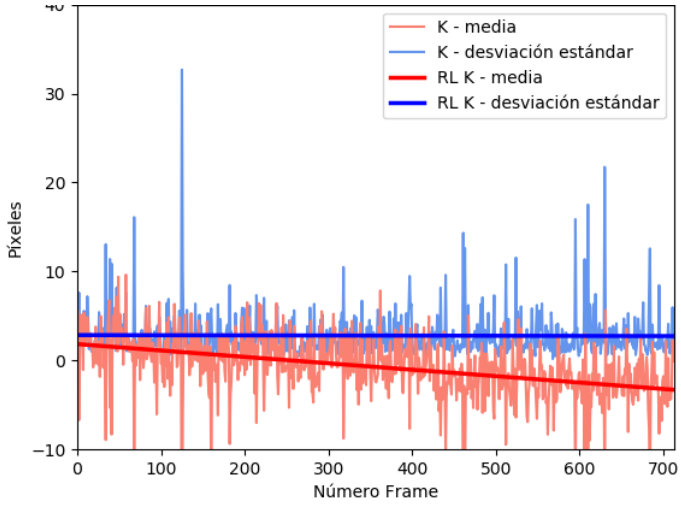

Figura 6.14: Ejemplo de pendiente negativa en el movimiento diagonal.

En el caso de pendiente estable, se observa que el número de casos extremos es mucho menor al de pendiente negativa. Esto implica que se realizan menos retornos al descubrimiento y se imcrementa los FPS totales del sistema.

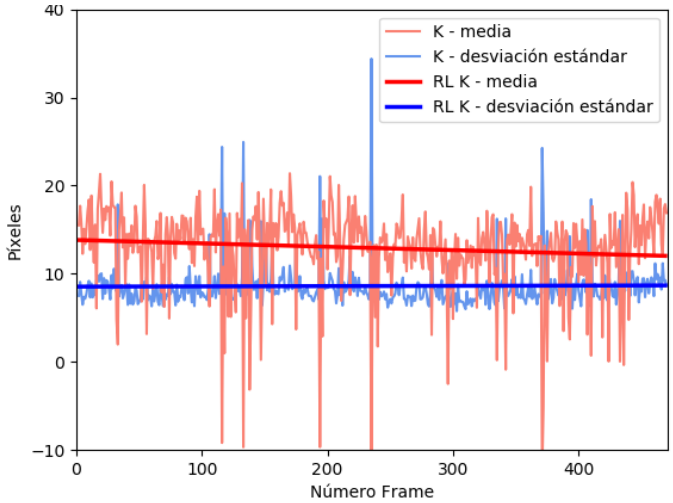

Figura 6.15: Ejemplo de pendiente estable en el movimiento diagonal.

En el caso de pendiente positiva, se observa que la desviación de las tendencias de la escala y la centralidad es menor que en el resto de los casos. Esto quiere decir que, aunque se amplíe el área la ROI seguimiento, se reduce el número de casos extremos. Esto indica que un pendiente positiva ayuda a la desviación de las tendencias en los movimientos diagonales.

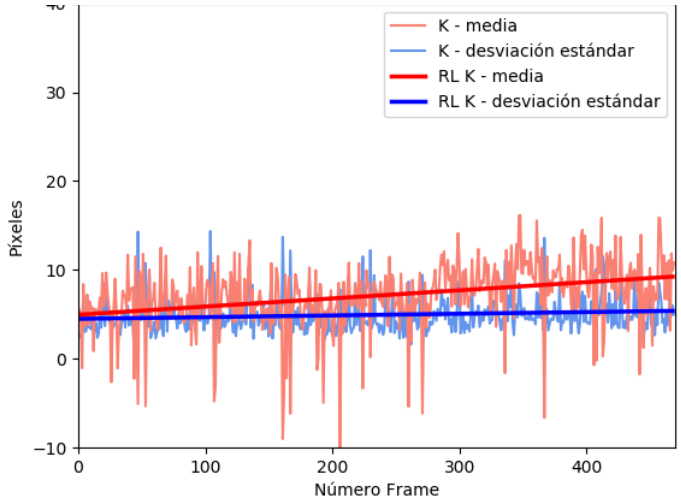

Figura 6.16: Ejemplo de pendiente positiva en el movimiento diagonal.

Las Figuras 6.17, 6.18, 6.19 y 6.20 muestran la tendencia y desviación de la escala y la tendencia y desviación de la centralidad para el movimiento diagonal respectivamente.

Respectivo a la tendencia temporal de la escala para el movimiento diagonal, se observa que el valor de delta igual 10 píxeles presenta los casos más estables. Su significado físico es que, en caso de que el incremento de la ROI sea bajo, el algoritmo de seguimiento no tiene margen de libertad y, con el paso del tiempo, el rendimiento del sistema empora. En caso contrario, si el incremento es alto, el algoritmo no es capaz de seguir correctamente la fuente transmisora. Por tanto, existe un valor de compromiso en el cual los valores de pendiente son mínimos. Además, para valores de delta alto se destaca que existe tendencia a estrechar a la fuente, esto es, un mayor número de casos son pendientes negativas.

Respectivo a la tendencia temporal de la centralidad para el movimiento diagonal, se observa que según se disminuye la delta, mayor serán los valores. Esto quiere decir que durante más libertad se deje a la ROI seguimiento, más excéntrico se volverá con el tiempo. Se debe a que el algoritmo no es capaz de situar correctamente al transmisor dentro de la ROI. Por tanto, desde el punto de vista de la centralidad, es mejor un valor bajo de delta.

Por tanto, para el movimiento diagonal se busca un valor mínimo de delta por la centralidad y un valor intermedio por la escala.

Además, para la distancia de 1 metro se observa que los valores de la tendencia temporal son mayores a la distancia de 2 metros. Esto quiere decir que durante más

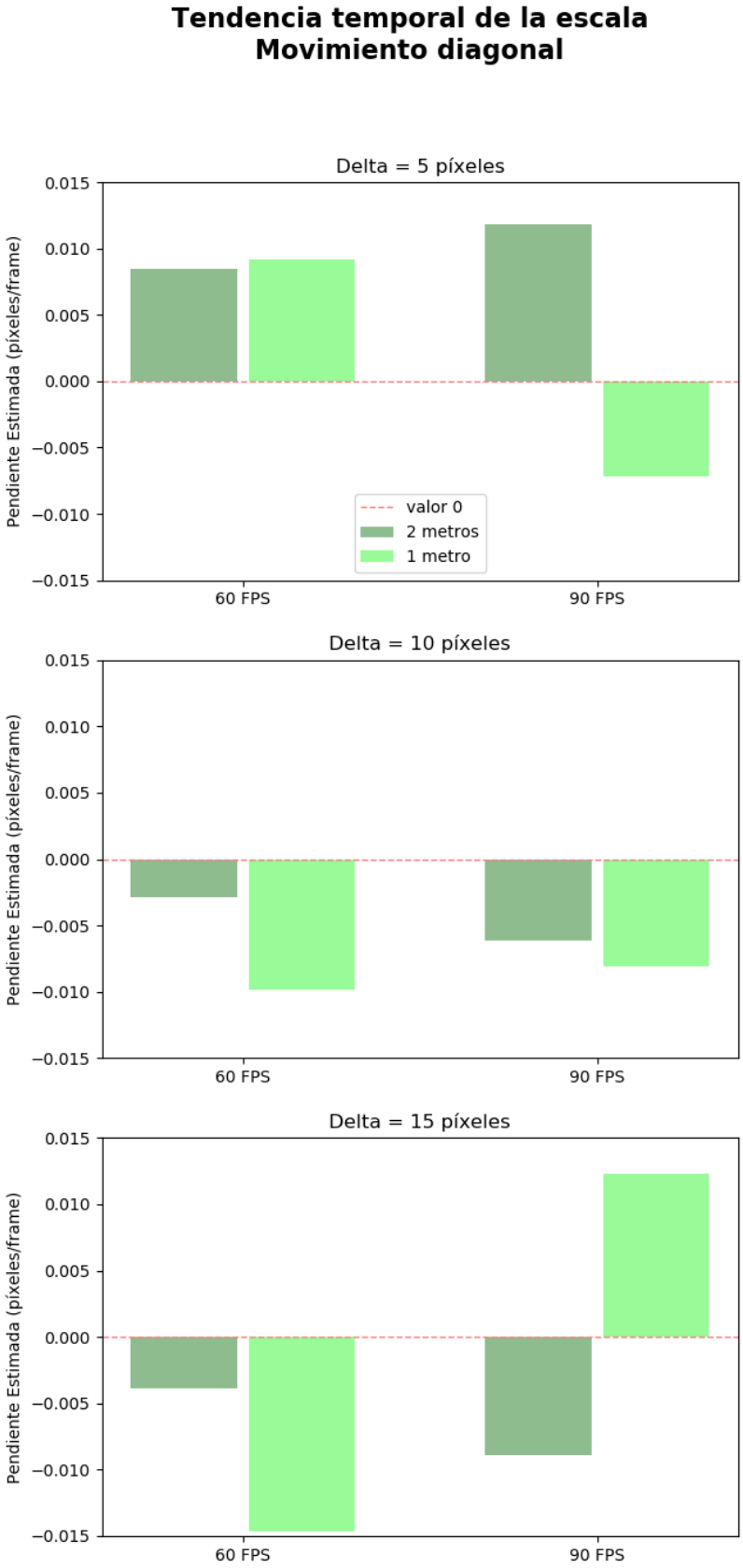

Figura 6.17: Tendencia temporal de la escala para el movimiento diagonal.

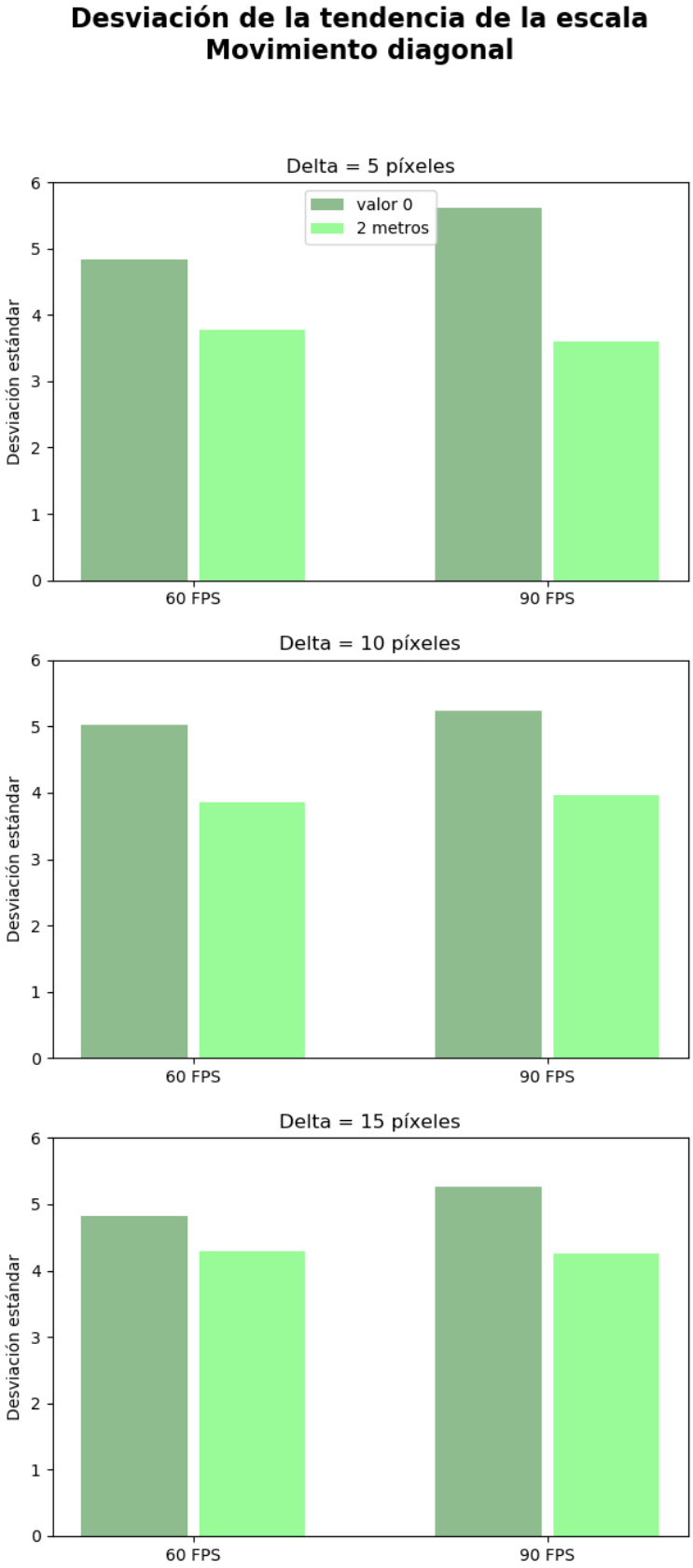

Figura 6.18: Desviación de la tendencia de la escala para el movimiento diagonal.

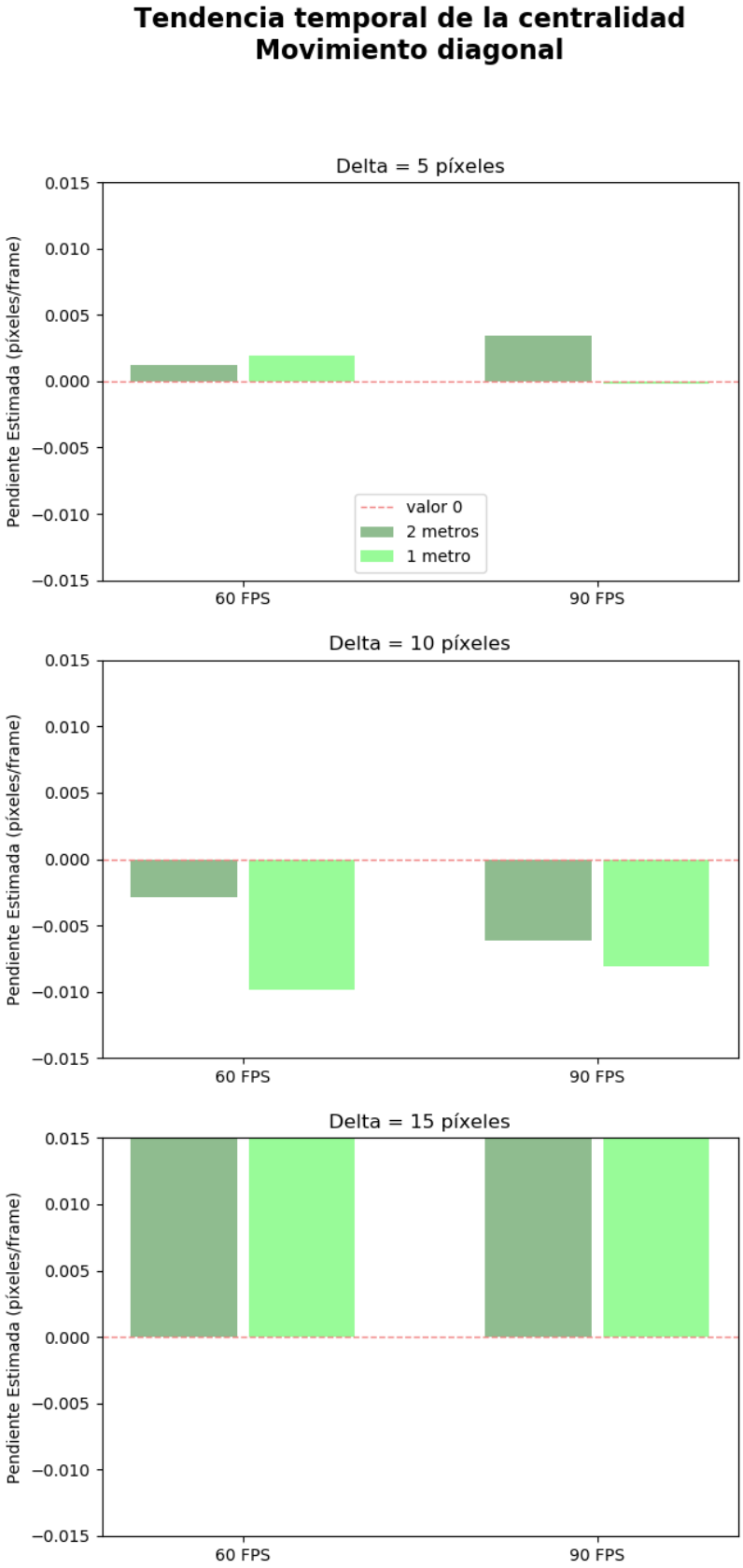

Figura 6.19: Tendencia temporal de la centralidad para el movimiento diagonal.

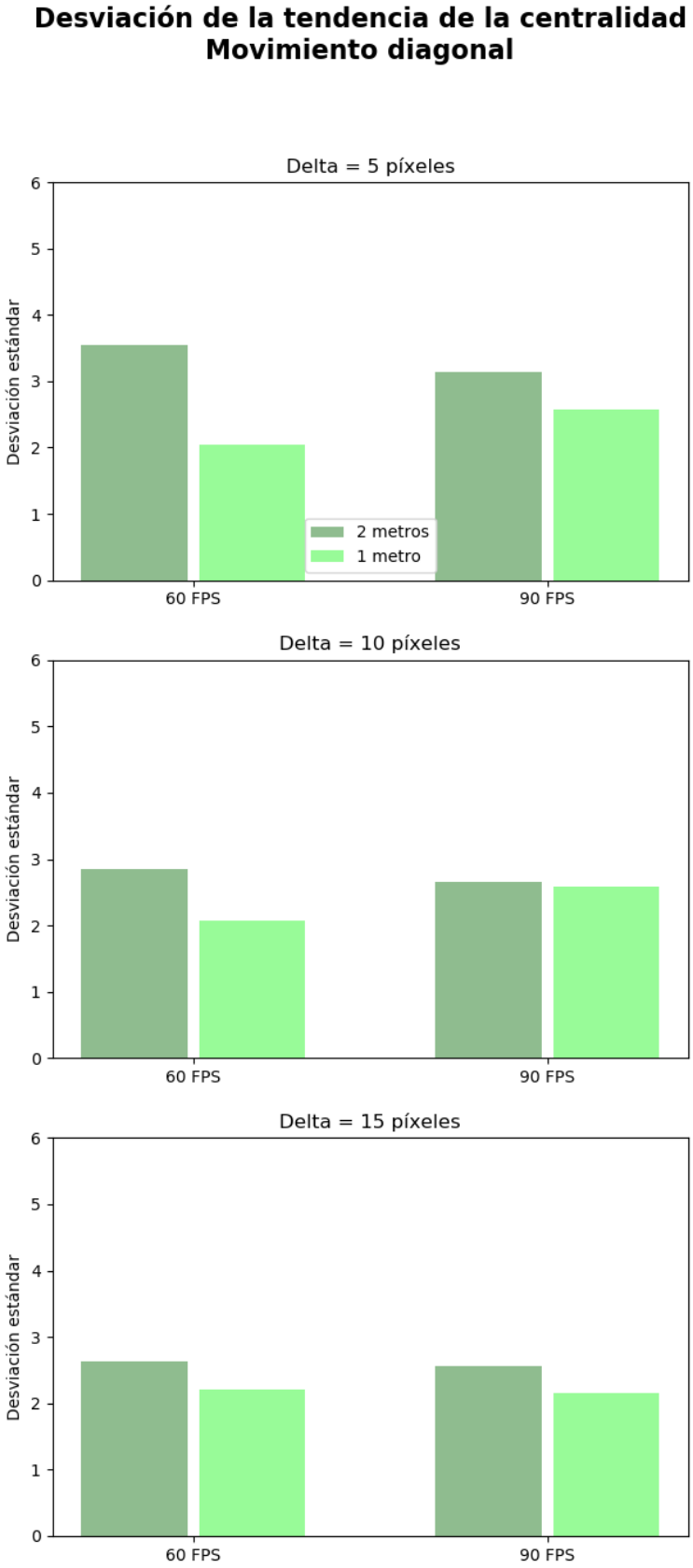

Figura 6.20: Desviación de la tendencia de la centralidad para el movimiento diagonal.

cerca se encuentra el objeto, la proyección del transmisor en la imagen presenta un área en píxeles mayor y, por ende, existe menos probabilidad de pérdida de centralidad. Por otro lado, las desviaciones de las tendencias disminuyen al aumentar la distancia, es decir, presentan menor ruido. Este fenómeno es provocado por la misma razón.

Por otro lado, no existen evidencias con los resultados obtenidos en las condiciones del experimento, la velocidad del transmisor, de que en el movimiento diagonal exista una mejora en el seguimiento con el incremento de la tasa de captura de la cámara.

Finalmente, recordar, como se indicó en las observaciones de la Figura 6.14, que el seguimiento para el movimiento diagonal tiene detacamente un peor rendimiento que para al movimiento lateral. Se puede observar por los valores de las tendencias de las gráficas. Esto quiere decir que la ROI seguimiento se contrae o expande a mayor velocidad, y se descentra con facilidad. Además, la desviación de la tendencia es mucho mayor en este movimiento. Por tanto, se puede concluir que el movimiento diagonal supone un tiempo de retorno al descubrimiento mucho menor.

6.2.1.3. Movimiento frontal

En las Figuras 6.14, 6.15 y 6.16 se muestran algunos ejemplos de la tendencia del algoritmo de seguimiento para el movimiento frontal.

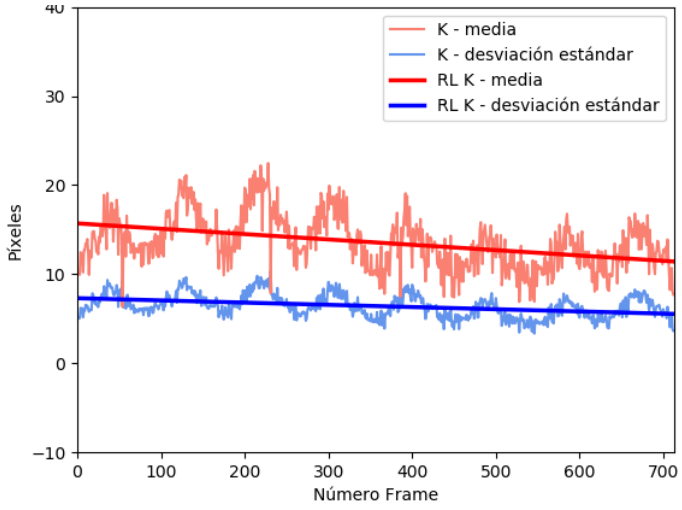

Figura 6.21: Ejemplo de pendiente negativa en el movimiento frontal.

En estas figuras se observa una periodicidad debida al cambio de escala de la fuente proyecta en la imagen (el transmisor se acerca y se aleja continuamente). Esto implica que el algoritmo de seguimiento se adapta a estos cambios de escala como se indicó en

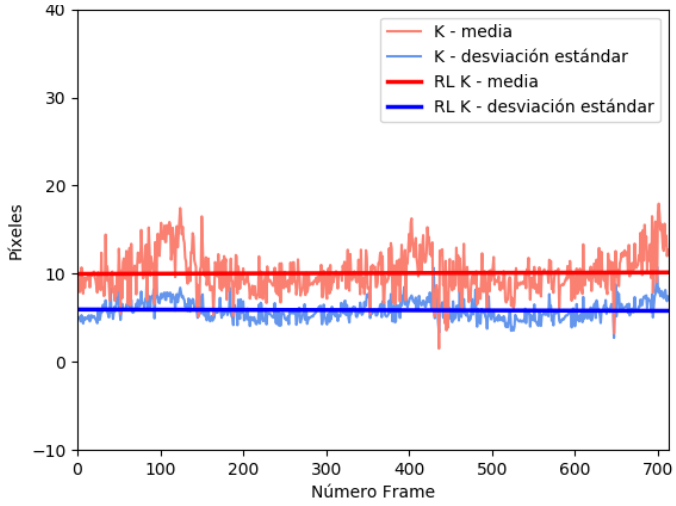

Figura 6.22: Ejemplo de pendiente estable en el movimiento frontal.

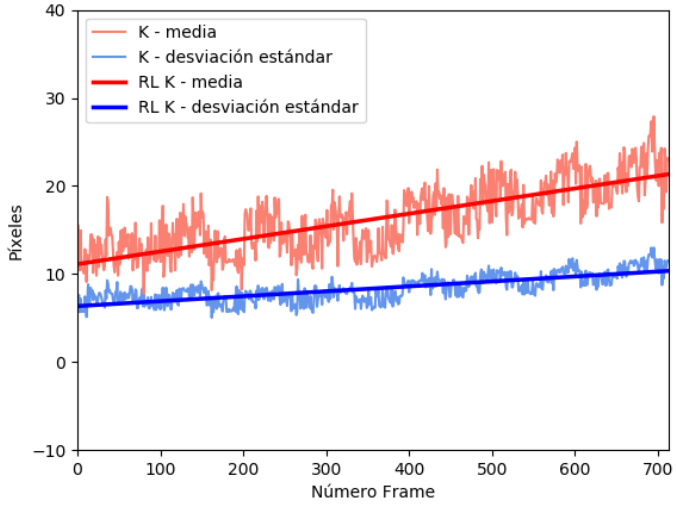

Figura 6.23: Ejemplo de pendiente positiva en el movimiento frontal.

el análisis realizado a los diferentes algoritmos de seguimiento.

En el caso de las Figuras 6.21 y 6.23, el vídeo está grabado a 90 FPS, y dado que el movimiento se repite cada 100 frames aproximadamente, se puede estimar, analizando la periodicidad de la señal, que la velocidad del movimiento del transmisor es $100 frames/90 FPS \approx 1{,}11 m/s$. Lo cual coincide con la velocidad seleccionada para su movimiento. Por otro lado, se puede observar que la señal es triangular. Esto se debe a que el movimiento del transmisor, producida por el ciclo de trabajo de la señal PWM, se modula con una señal triangular.

Dada esta periodicidad,la tendencia temporal tiene la misma interpretación que los otros movimientos. Sin embargo, la interpretación de la desviación de la tendencia no se puede analizar de forma directa. Debido a esto, solo se analiza la tendencia temporal.

En la Figura 6.21 (pendiente negativa) se puede observar que la ROI de seguimiento tiende a reducirse, aunque el objeto no ha variado en tamaño.

En la Figura 6.22 (pendiente estable) se puede observar dos cosas. Primero, que la velocidad es menor (dado que se mostrando el ejemplo del vídeo 1) y que, por tanto, el periodo de la señal es mayor. Además, en este caso, el sistema tiende a mantenerse estable a pesar del cambio de escala, su tendencia temporal es prácticamente constante.

En la Figura 6.23 (pendiente positiva) se puede que la ROI de seguimiento crece muy rápido, lo que supone que el sistema alcanza casos extremos mucho más rápido que obligan al retorno del descubrimiento.

Por otro lado, la desviación de la tendencia no se calcula en este caso, se observa que la variación entre las ROIs de seguimiento entre *frame* consecutivos es considerablemente pequeña.

Las Figuras 6.24 y 6.25 presentan los valores obtenidos de las tendencias temporales de la escala y la centralidad para el movimiento frontal.

En este caso se puede observar que cuanto mayor sea el incremento en delta (en píxeles), mayor será el valor de la pendiente. Esto significa que el significa tiende a alcanzar casos extremos con mayor rapidez para incrementos de la ROI seleccionada más grandes. Por tanto, una delta baja es de interés en este movimiento.

Por otra parte, para el vídeo 1, la magnitud de las tendencias son menores respecto al resto de casos. Esto es debido a que en vídeo el transmisor se mueve a una velocidad inferior. De aquí se deriva que cuanto menor es la velocidad, mejor será el comportamiento del sistema.

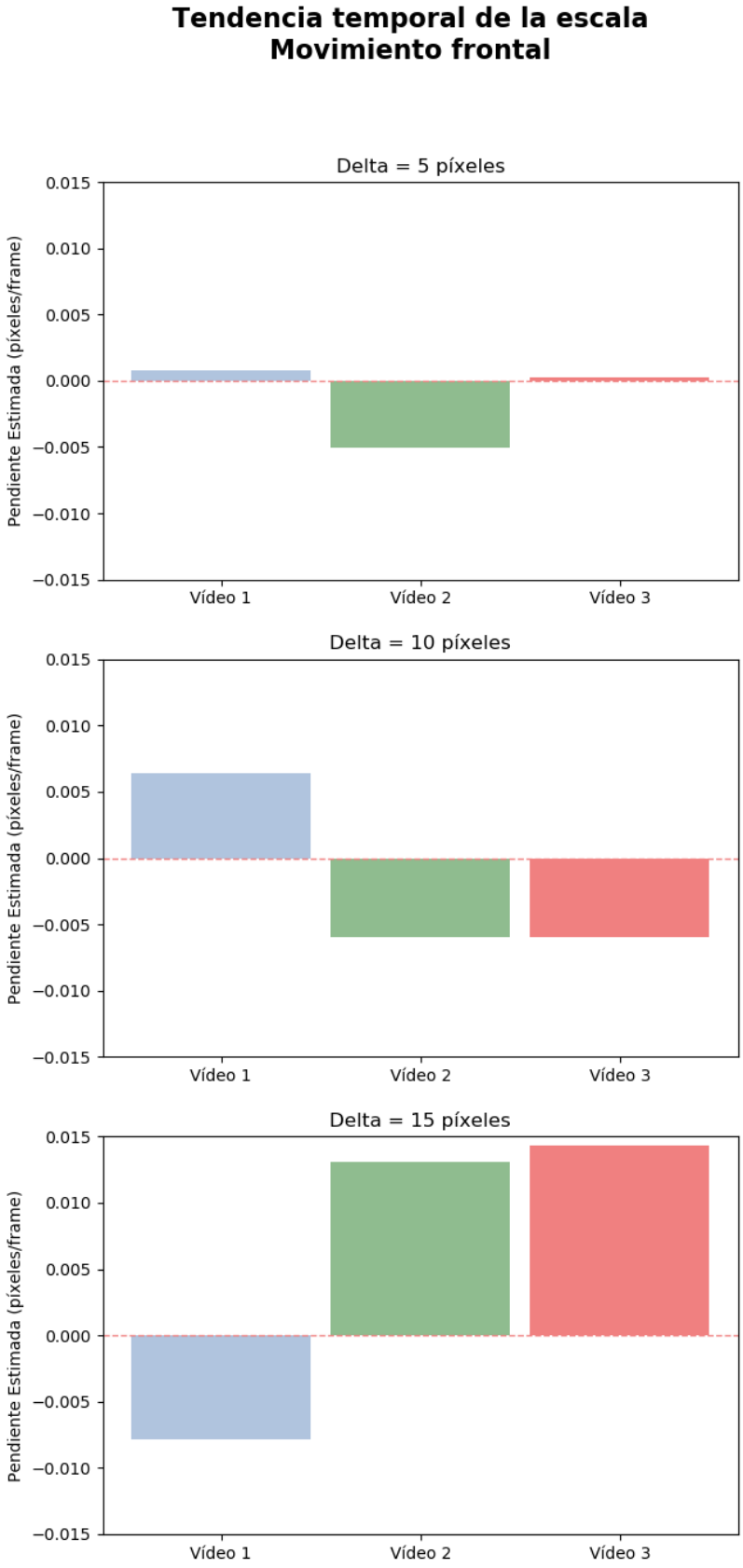

Figura 6.24: Tendencia temporal de la escala para el movimiento frontal.

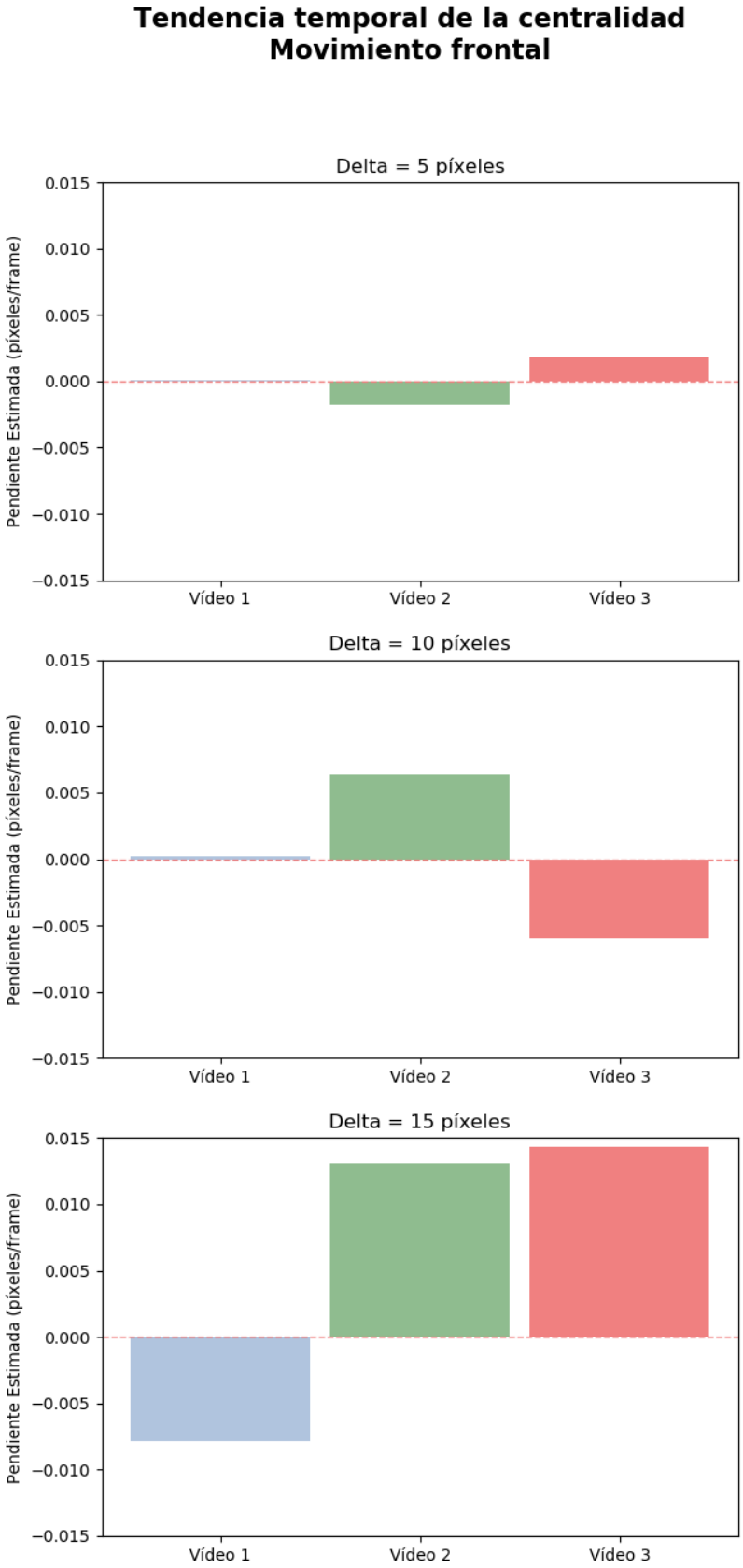

Figura 6.25: Tendencia temporal de la centralidad para el movimiento frontal.

6.2.1.4. Prueba de Student

En el capítulo 5, se propuso la prueba de Student para analizar si existe o no dependencia temporal de la escala y la centralidad conforme evoluciona el procesamiento de la captura. Este análisis es posible dado que el movimiento del transmisor es conocido y periódico. El procedimiento explicado en el capítulo de implementación consiste en obtener el parámetro T_0 y los valores críticos $T_{\pm\alpha/2,n-2}$, para una confianza del 95 % y el número de muestras que definen los grados de libertad.

En caso de que el parámetro T_0 se encuentre entre $T_{\alpha/2,n-2}$ y $T_{-\alpha/2,n-2}$, no se rechaza la hipótesis nula, la independencia temporal, y se cumple que el algoritmo de seguimiento es estable. En caso contrario, si se encuentra fuera de los valores críticos, se rechaza la hipótesis nula y se acepta la hipótesis de contraste, existe dependencia temporal, y el algoritmo de seguimiento no estable y tiende a diverger a los casos extremos.

Las tablas 6.2, 6.3 y 6.4 recogen los resultados de la prueba de Student para cada tipo de movimiento. Mencionar que "R" hace referencia a que rechaza la hipótesis nula, mientras que "NR" no rechaza la hipótesis nula.

Tabla 6.2: Prueba de Student para el movimiento lateral.

		Delta		
Distancia	FPS	5	10	15
2 metros	60	R	R	R
	90	R	R	R
1.3 metros	60	R	R	NR
	90	R	R	R

Tabla 6.3: Prueba de Student para el movimiento diagonal.

		Delta		
Distancia	FPS	5	10	15
2 metros	60	R	NR	R
	90	R	R	R
1.3 metros	60	R	R	R
	90	R	R	R

De estas tablas se observa que en el 91 % de los casos se rechaza la hipotesis nula, lo que implica que se debe retornar al descubrimiento después de un tiempo. Esto se debe a que la ROI seguimiento se deteriora con el tiempo. Además, rechazar la hipótesis

Tabla 6.4: Prueba de Student para el movimiento frontal.

	Delta		
	5	10	15
Vídeo 1: 0.5m/s y 1.3 m	R	NR	R
Vídeo 2: 1.1m/s y 1.3 m	R	R	R
Vídeo 3: 1.1m/s y 2 m	R	R	R

nula, junto a que los valores residuales siguen una distribución gaussiana, nos indica que, con los datos obtenidos, el uso de la regresión lineal es adecuado.

Por otro lado, existen 3 de los 33 casos en los cuales no se rechaza la hipótesis nula y se acepta la hipótesis de contraste. Esto implica que con un 95 % de confianza podemos asegurar, a priori, que la pendiente es cero, la ROI seguimiento permanece constante, y no se tiene que volver al descubrimiento. Sin embargo, destacar que, con los datos obtenidos, no se encuentra relación contundente entre los parámetros de los casos de no rechazo, por lo que no se puede resaltar ningún parámetro que condicione este comportamiento.

Capítulo 7

Conclusiones

En este Trabajo Fin de Grado "Diseño de un sistema OCC para dispositivos wearables" se ha diseñado, implementado y evaluado el sistema de comunicación OCC planteado en esta memoria.

Las comunicaciones por luz visible VLC constituyen hoy en día una solución viable al actual y creciente incremento de los dispositivos interconectados. Sin embargo, el uso de fotorreceptores implica un coste e impacto mayor en la implementación y dificulta la penetración de esta tecnología para el usuario común.

Como solución a este problema, surge la tecnología OCC que utiliza las cámaras como receptores ópticos, aprovechándose de todas las ventajas de las comunicaciones VLC y facilitando la entrada de estos dispositivos en el mercado.

No obstante, la tarea de incorporar estos sistemas a dispositivos electrónicos portables, que se mueven libremente, presenta aún una serie de dificultades relacionadas a las cuales se pretende dar solución con el trabajo desarrollado en esta memoria.

En definitiva, este trabajo presenta una solución al problema de movilidad y probabilidad, que analiza las técnicas de descubrimiento y seguimiento de la fuente, y propone una arquitectura novedosa. La solución propuesta aprovecha las características de la fuente como elemento de iluminación activo, y de las cámaras como sistemas capaces de regular su sensibilidad a la luz mediante el tiempo de exposición.

En este trabajo se implementa una solución atacando los subsistemas de descubrimiento, encargado de detectar el transmisor, y seguimiento, encargado de seguirlo. Este conjunto permite que el sistema procese capturas con un mayor número de FPS como se comprobó en la evaluación del sistema.

Esta solución permite que para una misma restricción de fotogramas por segundos, se precisen menos recursos *hardware*, lo que posibilita que exista un mayor número de dispositivos donde la solución propuesta pueda ser desplegada.

Por otra parte, en la actualidad, la mayoría de las cámaras de propósito general integradas en los dispositivos electrónicos utilizan el mecanismo de adquisición *Rolling Shutter*, por lo que la imagen se obtiene fila a fila. Esto implica que los sistemas OCC son sensibles a la desincronización. Por esto, en la solución propuesta se diseña e implementa una técnica que mantiene el dato durante dos o más tiempos de frames. De esta forma, no se requieren técnicas de sincronización complejas y se evita la mezcla *intra-frames* de los símbolos.

Al final de este trabajo, se diseñó un entorno y una estrategia de evaluación para validar el sistema. La estrategia de evaluación aquí propuesta, basadas en los parámetros K, puede ser utilizada para el análisis de otras arquitecturas de seguimiento (incluso aquellas no ligadas a la tecnología OCC).

Los resultados obtenidos permiten indicar que el sistema diseñado e implementado cumple con las siguientes características:

- el transmisor se puede integrar con un dispositivo wearable que se mueva con una cierta libertad.
- el receptor detecta a la fuente transmisora en movimiento con una sensbilidad mayor al 97.5 % y un tiempo de ejecución menor a 0.1 segundos para las condiciones de este experimento.
- el receptor sigue a la fuente transmisora en movimiento con éxito.

Con los resultados obtenidos se puede concluir que se han conseguido todos los objetivos declarados para ese trabajo en el capítulo de Introducción.

Finalmente, cabe señalar que este Trabajo Fin de Grado supone una base teórica-práctica para futuros trabajos. Las líneas futuras de trabajo planteadas se centran en la implementación del sistema final con todas las etapas (demodulación y decodificación) ya descritas en el diseño y tenidas en cuenta en el diseño e implementación del transmisor. Además, el sistema está sujeto a modificaciones siempre y cuando se cumpla el esquema desarrollado en este trabajo. Por tanto, las líneas planteadas son:

- Evaluar el impacto de otros tipos de algoritmos de generación de *proposals*.

- Evaluar el impacto de otro tipo de clasificador, como puede ser una red neuronal convolucional entre otras propuestas.
- Evaluar el impacto de otros tipos de algoritmos de seguimiento.
- Evaluar el comportamiento del sistema con transmisores interferentes.
- Implementar los sistemas de demodulación y decodificación.
- Diseñar e implementar el sistema de generación de energía del transmisor.
- Integrar y evaluar los resultados de la solución planteada para el receptor en dispositivos electrónicos móviles.
- Evaluar la solución planteada en otros entornos, como el urbano nocturno.

Acrónimos

ADC Analog to Digital Converter. 11, 12, 55

COITT Colegio Oficial de Ingenieros Técnicos de Telecomunicación. 152, 155, 156

FPS Frames Per Second. 10, 14, 19, 34, 36, 94, 99, 102, 108–112, 115, 120, 121, 128, 135

IEEE Institute of Electrical and Electronics Engineers. 3

IGIC Impuesto General Indirecto Canario. 158

IoT Internet of Things. 3, 9

LED Light-Emitting Diode. 3, 5, 9, 14, 19, 27–35, 39, 40, 43–49, 52, 55, 56, 58, 63, 65, 69, 83

OCC Optical Camera Communications. 4–7, 10, 14–16, 19, 25, 29, 32, 38, 135, 136

OOK On-Off Keying. 4, 27, 35, 46

OWC Optical Wireless Communications. 3

RGB Red-Green-Blue. 5, 21, 32, 44, 47

ROI Region Of Interest. 16, 17, 19, 21, 22, 37–41, 60, 63–65, 67–75, 77, 78, 81–84, 94, 95, 97–99, 101–104, 106, 112–115, 120–122, 127, 129, 132, 133

TFG Trabajo de Fin de Grado. 7, 161

UPSOOK Undersampled Phase Shift OOK. 4, 5

VLC Visible Light Communications. 3, 9, 10, 14, 135

Bibliografía

[1] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, 2013.

[2] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[3] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Proceedings - International Conference on Pattern Recognition*, 2010.

[4] "Raspberry Pi Camera spectral response curves." [Online]. Available: https://khufkens.github.io/pi-camera-response-curves/

[5] N. T. Le, M. A. Hossain, and Y. M. Jang, "A survey of design and implementation for optical camera communication," *Signal Processing: Image Communication*, 2017.

[6] "IEEE Standard for Local and metropolitan area networks–Part 15.7: Short-Range Optical Wireless Communications - Redline," *IEEE Std 802.15.7-2018 (Revision of IEEE Std 802.15.7-2011) - Redline*, pp. 1–670, 2019.

[7] Z. Ghassemlooy, P. Luo, and S. Zvanovec, "Optical Camera Communications," 2016.

[8] N. T. Le, M. A. Hossain, and Y. M. Jang, "A survey of design and implementation for optical camera communication," *Signal Processing: Image Communication*, 2017.

[9] M. Shahjalal, M. K. Hasan, M. Z. Chowdhury, and Y. M. Jang, "Future Optical Camera Communication Based Applications and Opportunities for 5G and beyond," in *1st International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2019*, 2019.

[10] P. Luo, Z. Ghassemlooy, H. Le Minh, X. Tang, and H. M. Tsai, "Undersampled phase shift ON-OFF keying for camera communication," in *2014 6th International Conference on Wireless Communications and Signal Processing, WCSP 2014*, 2014.

[11] V. P. Rachim, Y. Jiang, H.-S. Lee, and W.-Y. Chung, "Demonstration of long-distance hazard-free wearable EEG monitoring system using mobile phone visible light communication," *Optics Express*, 2017.

[12] S. Teli, W. A. Cahyadi, and Y. H. Chung, "Optical Camera Communication: Motion over Camera," *IEEE Communications Magazine*, 2017.

[13] C. Jurado-Verdu, V. Guerra, J. Rabadan, R. Perez-Jimenez, and P. Chavez-Burbano, "RGB Synchronous VLC modulation scheme for OCC," in *2018 11th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2018*, 2018.

[14] C. Jurado-Verdu, V. Matus, J. Rabadan, V. Guerra, and R. Perez-Jimenez, "Correlation-based receiver for optical camera communications," *Optics Express*, vol. 27, no. 14, p. 19150, 7 2019.

[15] T. Nguyen, N. T. Le, and Y. M. Jang, "Asynchronous Scheme for Optical Camera Communication-Based Infrastructure-to-Vehicle Communication," *INTERNATIONAL JOURNAL OF DISTRIBUTED SENSOR NETWORKS*, 2015.

[16] T. Nagura, T. Yam, M. Katayama, T. Yendo, T. Fujii, and H. Okada, "Tracking an LED array transmitter for visible light communications in the driving situation," in *2010 7th International Symposium on Wireless Communication Systems*. IEEE, sep 2010, pp. 765–769. [Online]. Available: http://ieeexplore.ieee.org/document/5624361/

[17] H. C. N. Premachandra, T. Yendo, M. P. Tehrani, T. Yamazato, H. Okada, T. Fujii, and M. Tanimoto, "High-speed-camera image processing based LED traffic light detection for road-to-vehicle visible light communication," in *2010 IEEE Intelligent Vehicles Symposium*. IEEE, jun 2010, pp. 793–798. [Online]. Available: http://ieeexplore.ieee.org/document/5548097/

[18] A. Islam, M. T. Hossan, and Y. M. Jang, "Convolutional neural networkscheme–based optical camera communication system for intelligent Internet of vehicles," *International Journal of Distributed Sensor Networks*, vol. 14, no. 4, p. 155014771877015, apr 2018. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1550147718770153

[19] R. Girshick, J. Donahue, T. Darrell, U. C. Berkeley, and J. Malik, "R-CNN," *1311.2524v5*, 2014.

[20] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[23] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *BMVC 2006 - Proceedings of the British Machine Vision Conference 2006*, 2006.

[24] B. Babenko, M. H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[25] M. Tang, B. Yu, F. Zhang, and J. Wang, "High-Speed Tracking with Multi-kernel Correlation Filters," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[26] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[27] "Adafruit DotStar Digital LED Strip - White 144 LED/m." [Online]. Available: https://www.adafruit.com/product/2242

[28] "STM32L432KC." [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32l432kc.html

[29] "Raspberry Pi Camera Module V2 8-megapixel." [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/

Parte II

Pliego de Condiciones

Pliego de Condiciones

A continuación se especifican los requisitos técnicos de este Trabajo de Fin de Grado.

Requisitos hardware

El hardware requerido queda recogido en la Tabla 7.1.

Tabla 7.1: Requisitos Hardware.

Componente	**Modelo**	**Fabricante**	**Un.**
Módulo LED			
Tiras Addressable LED RGB	DotStar 144LEDs/m[27]	Adafruit	1
Módulo de comunicaciones			
Microcontrolador	NUCLEO-L432KC[28]	STM	1
Receptor			
Cámara	Pi Camera Module V2 8MP Pro[29]	Raspberry	1
Ordenador			
Ordenador	Acer Aspire 5 A515-51G	Lenovo	1

Requisitos software

El software requerido queda recogido en la Tabla 7.2.

Tabla 7.2: Requisitos Software.

Sofware	**Versión**	**Fabricante**
Sistema Operativo	Windows 10	Microsoft Windows
Arm Mbed OS	5.6	Arm
Python	3.6	Python
OpenCV	4.01	OpenCV

Esquemáticos

El esquemático del módulo del transmisor se muestra en la Figura **??**.

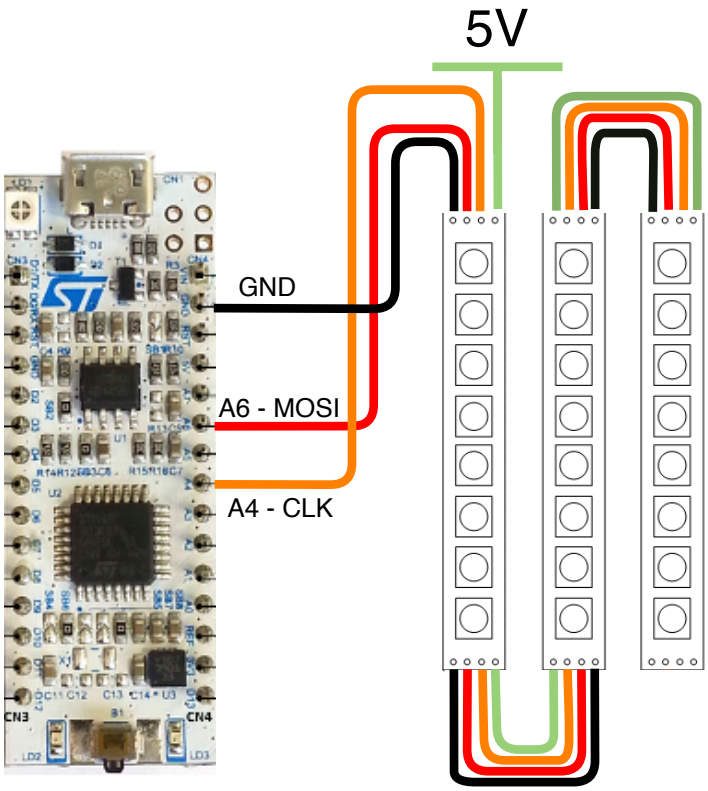

Figura 7.1: Conexionado del transmisor.

Parte III

Presupuesto

Presupuesto

Introducción

En este capítulo se estimaron los gatos generados por el Trabajo Fin de Grado presentado en esta memoria. El presupuesto se divide en las siguientes partes:

- Tarifa de honorarios por tiempo empleado.
- Amortización y coste de los equipos y materiales empleados.
  - Coste del material hardware y amortizaciones.
  - Amortización del material software.
- Coste de acceso a Internet.
- Redacción de la documentación.
- Derechos de visado.
- Gastos de tramitación y envío.

Tarifa de honorarios por tiempo empleado

Este concepto contabiliza los gastos correspondientes a la mano de obra. Para realizar este cálculo, se propone la siguiente fórmula[1]:

$$H = C_t \times (75 \times H_n) + C_t \times (95 \times H_e)$$

Siendo:

- H: los honorarios totales por tiempo dedicado.

[1]Según recuerda el Ministerio de Economía y Hacienda, y siguiendo directivas europeas y con aplicación al año 2011, se deben eliminar los baremos orientativos de honorarios que los colegios tradicionalmente venían publicando en sus plataformas. Es por ello que se utilizarán coeficientes estimativos de acuerdo a la condición de ingeniero superior recién titulado.

- H_n: los honorarios en jornada laboral normal.
- H_e: los honorarios fuera de la jornada laboral normal.
- C_t: el factor de corrección según el número de horas trabajadas.

Para la realización del presente TFG se han invertido un total de 300 horas, repartidas en jornadas de 6 horas y media durante un periodo de cincuenta días. Todas ellas se han realizado dentro del horario normal, por lo que el número de horas especiales no son consideradas. Además, de acuerdo a lo establecido por el Colegio Oficial de Ingenieros Técnicos de Telecomunicación (COITT), el factor de corrección C_t a aplicar para 300 horas trabajadas es de 0.60, tal y como se puede comprobar en la Tabla 7.3.

Horas	**Factor de Corrección**, C_t
Hasta 36	1
de 36 a 72	0.90
de 72 a 108	0.80
de 108 a 144	0.70
de 144 a 188	0.65
de 188 a 360	0.60
de 360 a 510	0.55
de 510 a 720	0.50
de 720 a 1080	0.45

Tabla 7.3: Factores de corrección para el número de horas empleadas.

Finalmente, obtenemos el coste total de los honorarios, que asciende *TRECE MIL QUINIENTOS EUROS*:

$$H = 0{,}6 \times 75 \times 300\,€ = 13.500\,€ \tag{7.1}$$

Amortización y coste de los equipos y materiales empleados

Dentro de este concepto se considera tanto la amortización del hardware como del software empleado en la realización del trabajo presentado. De este modo, se estipula el coste de amortización para un período de 3 años, utilizando un sistema de amortización lineal o constante. En este sistema, se supone que el inmovilizado material se deprecia

de forma constante a lo largo de su vida útil. La cuota de amortización anual se calcula haciendo uso de la siguiente fórmula:

$$C = \frac{Vad - V}{N} \tag{7.2}$$

Donde:

- **C**: cuota de amortización anual.
- **Vad**: valor de la adquisición.
- **V**: valor residual.
- **N**: número de años de vida útil de la adquisición.

Siendo el valor residual el valor teórico que se supone tendrá el elemento en cuestión después de su vida útil, teniendo en cuenta los índices de depreciación actual. En el caso del hardware y del software son 3 años (al 33 % de depreciación máximo por año).Sin embargo, debido a que el trabajo se ha elaborado en un periodo inferior a 3 años, que es el periodo en que se calcula la amortización de material hardware, se realizará una amortización equiparable al período de duración del mismo. Teniendo esto en cuenta, la depreciación del valor por los tres meses será del (33 % / 12meses $\times$ 3meses = 8,25 %). Resultando que el valor residual es $V_{\text{res}} = V_{\text{ad}} \times 0{,}0825$. Nótese que tanto el tiempo de uso de los aparatos, como el tiempo de vida útil será $t_{\text{uso}} = 3$ meses y $t_{\text{util}} = 3$ años.

Coste del material hardware y amortizaciones.

Para las amortizaciones se utilizó la Ecuación 7.2. En la Tabla 7.4 se muestran los costes y amortizaciones del hardware.

Por lo tanto, el coste total de hardware asciende a la cantidad de *MIL NOVECIENTOS OCHENTA Y OCHO CON CINCUENTA DE EURO.*

Tabla 7.4: Precios y costes de amortización del hardware.

Descripción	**Ud**	V_{ad}	V_{res}	**Total**
Desglose: Material de Trabajo				
Acer Aspire 5 A515-51G	1	599,00 €	49,5 €	183,5 €
Desglose: Aparatos de medición y herramientas				
Generador de Funciones	1	2000 €	175 €	608,33 €
Osciloscopio	1	3500 €	288,75€	1070,41 €
Soldador (con estaño para soldadura)	1	45 €	3,71 €	13,76 €
Desglose: Elementos del sistema transmisor (No amortizables)				
Micro STM32L432KC	1	13 €	-	13 €
Tiras Addressables LED	1	43 €	-	43 €
Tela, velcro, pegamento y goma eva	1	10 €	-	10 €
Plancha de metacrilato	1	15 €	-	15 €
Corte láser	1	1 €	-	1 €
Cables y Conectores	1	1,5 €	-	1,5 €
Desglose: Elementos del sistema receptor (No amortizables)				
Raspberry Pi Camera V2	1	29 €	-	29 €
Total:				1988,5 €

Amortización del material software

El coste total del software utilizado es la suma de las herramientas software empleadas para la realización del trabajo (Tabla 7.5). De esta manera se describe cada una de las herramientas utilizadas y el coste supuesto para su utilización usando la fórmula de amortización anteriormente citada.

Tabla 7.5: Precios y costes de amortización del software.

Descripción	V_{ad}	V_{res}	**Total**
Windows 10	0,00€	0,00 €	0,00 €
MbedOS	0,00 €	0,00 €	0,00 €
Python 3.6	0,00 €	0,00 €	0,00 €
OpenCV	0,00 €	0,00 €	0,00 €
OverLeaf (entorno de LaTeX)	0,00 €	0,00 €	0,00 €
DrawIO	0,00 €	0,00 €	0,00 €
Total			0,00 €

Por tanto, el coste total de software asciende a *CERO EUROS.*

Coste de acceso a Internet

Para la conexión a Internet se dispone de una solución Fibra Óptica a 600Mbps cuyo coste mensual es de 47,99 euros. Debido a que se ha usado dicha conexión durante cada una de las etapas de creación del mismo (2 meses), el coste de acceso a Internet se traduce a un total de *NOVENTA Y CINCO CON NOVENTA Y OCHO DE EURO.*

Redacción de la documentación

Para calcular el valor monetario de la redacción del trabajo se aplicará la fórmula siguiente:

$$R = 0,07 \times P \times C_n \tag{7.3}$$

Siendo:

- **R**: coste de la redacción.
- **P**: presupuesto.
- C_n: coeficiente de ponderación en función del presupuesto

El valor de P se obtiene sumando los costes de las secciones anteriores tal y como muestra la tabla 7.6. El coeficiente de ponderación C_n para presupuestos menores de 30.050,00€ viene definido por el COITT con un valor de 1.

Tabla 7.6: Precios y costes de la ejecución del trabajo más las amortizaciones y el acceso a Internet.

Concepto	**Coste**
Tarifa de honorarios por tiempo empleado	13500,00 €
Costes y amortizaciones del hardware	1988,5 €
Amortización del material software	0,00 €
Coste de acceso a Internet	95,98 €
Total	**15584,48 €**

De este modo, se obtiene que:

$$R = 0,07 \times 15\,584.48 € = 1090.91 €$$ (7.4)

Al coste de redacción obtenido hasta el momento se le deben añadir otros gastos,

quedando el importe final de redacción del trabajo como se detalla en la Tabla 7.7. Por lo tanto, la redacción del trabajo asciende a un total de *MIL OCHENTA Y SIETE CON TREINTA Y CINCO CÉNTIMOS DE EURO*.

Tabla 7.7: Coste final de redacción del trabajo.

Concepto	**Coste**
Redacción del trabajo	1090,91 €
Papel de impresión	5,00 €
Impresión	25,00 €
Encuadernación	6,00 €
Total	**1126,91 €**

Derechos de visado

El COITT establece que los derechos de visado para proyectos técnicos en el año 2019 se calculan de acuerdo con la siguiente ecuación:

$$V = 0,006 \times P_1 \times C_1 + 0,003 \times P_2 \times C_2 \tag{7.5}$$

Siendo:

- **V**: coste del visado.
- P_1: presupuesto del proyecto.
- C_1: coeficiente reductor en función del presupuesto.
- P_2: presupuesto de ejecución material correspondiente a la obra civil.
- C_2: coeficiente reductor en función de P_2

El valor del coeficiente C_1 para proyectos de presupuesto inferior a 30.050,00€ es de 1. Al mismo tiempo el valor de P_2 es de 0€ ya que no se realiza ninguna obra civil. El valor del coeficiente se obtiene de la Tabla 7.8, mientras que el valor de P es el valor total del presupuesto, que se muestra en la Tabla 7.9.

Tabla 7.8: Tabla de coeficientes para el cálculo del visado.

Coste del presupuesto (€)	**Factor de Correlación (C)**
Hasta 30.050	1
Exceso de 30.050 hasta 60.101	0.9
Exceso de 60.101 hasta 90.151	0.8
Exceso de 90.151 hasta 120.202	0.7
...	

Tabla 7.9: Calculo total de presupuesto base para el cálculo del visado.

Concepto	**Coste**
Coste del tiempo empleado en la ejecución, amortización y acceso a Internet	15584,48 €
Redacción del trabajo	1126,91 €
Total	**16711,39 €**

Por lo tanto, el valor del visado será de:

$$V = 0,006 \times 16\,711.39\,€ = 100.26\,€ \tag{7.6}$$

Los costes de derechos de visado del trabajo ascienden a un total de *CIEN CON VEINTISÉIS CÉNTIMOS DE EURO*.

Gastos de tramitación y envío

Los gastos de tramitación y envío según la tarifa asciende a *SEIS EUROS* por cada documento visado de forma telemática.

Presupuesto antes de impuestos

Sumando todos los conceptos calculados hasta el momento, se obtiene el presupuesto, sin incluir los impuestos, que se muestra en la Tabla 7.10.

El presupuesto calculado, antes de incluir los impuestos, asciende a *DIECISÉIS MIL CIENTO NOVENTA CON OCHENTA Y CINCO CÉNTIMOS DE EURO*.

Tabla 7.10: Presupuesto total sin impuestos.

Concepto	**Coste**
Presupuesto base	16711,39 €
Derechos de visado	100,26 €
Gastos de tramitación y envío	6,00 €
Total	**16817,65 €**

Presupuesto incluyendo impuestos

Al presupuesto calculado anteriormente hay que incluirle un 6.5 % de Impuesto General Indirecto Canario (IGIC) obteniendo el coste del presupuesto final (Tabla 7.11).

Tabla 7.11: Presupuesto total.

Concepto	**Coste**
Total (sin IGIC)	16817,65 €
IGIC (6.5 %)	1093,15 €
Total	**17910,8 €**

El presupuesto total del trabajo "Diseño de un sistema OCC para dispositivos wearables", incluyendo impuestos, asciende a la cantidad de *DIECISIETE MIL NOVECIENTOS DIEZ CON OCHENTA CÉNTIMOS DE EURO*.

Las Palmas de Gran Canaria, a 12 de diciembre de 2019

Fdo: Antonio R. Mederos Barrera

Parte IV

Anexos

Anexo A

Contenidos adjuntos

En este Anexo se presenta el contenido del soporte físico adjunto a este documento, el cual incluye los siguientes archivos:

1. Memoria del TFG en formato PDF.
2. Código fuente utilizado tanto para la implementación como la evaluación del sistema. Este código se encuentra organizado en los siguientes directorios, dentro de la carpeta "codigos":
   a) **Receptor:** Incluye los códigos Python utilizados para la implementación del receptor y la evaluación *offline*.
   b) **Transmisor:** Incluye el código C++ utilizado para la implementación del transmisor.
   c) **Vídeos:** Incluye código bash usado para la obtención de los vídeos.
3. Vídeos para la evaluación en la carpeta "videos".
4. Resultados de la evaluación en la carpeta "resultados".