Universidad Zaragoza
1542

Proyecto Fin de Carrera

Captura de vídeos de alta resolución temporal utilizando exposición codificada y aprendizaje de diccionarios

Autor

Ana Serrano Pacheu

Directora: Belén Masiá Corcoy
Ponente: Dr. Diego Gutiérrez Pérez

Escuela de Ingeniería y Arquitectura
2014

Resumen

En el campo de la teoría de señal, el muestreo y la compresión siempre han sido puntos claves. El teorema de muestreo de Nyquist-Shannon es fundamental, especialmente en el campo de las telecomunicaciones. Este teorema establece que la reconstrucción exacta de una señal continua en banda base y limitada en banda a partir de sus muestras, es posible siempre y cuando su frecuencia de muestreo sea igual o superior al doble de su ancho de banda. Esto supone una barrera a la hora de muestrear ciertas señales de frecuencia elevada, ya que empiezan a aparecer limitaciones en el hardware necesario. Recientemente se ha estado desarrollando un nuevo campo en teoría y procesado de señal llamado *compressive sensing*. La teoría de *compressive sensing* establece que una señal puede reconstruirse perfectamente incluso cuando es muestreada a frecuencias menores que las establecidas por el teorema de Nyquist siempre que sea lo suficientemente dispersa (*sparse* según el término en inglés), esto es, que gran parte de los coeficientes en su representación en el dominio de alguna base sean cero, y el muestreo cumpla una serie de condiciones.

Para poder trabajar en este marco, es pues necesario transformar la señal a una base (también llamadas diccionarios) en la que sea *sparse*. Una vez se cuenta con este diccionario, y conociendo el patrón de muestreo, es posible reconstruir la señal original a partir de una versión muestreada por debajo del límite de Nyquist de la misma mediante la resolución de un problema de minimización L1.

Uno de los campos de aplicación de esta teoría es la captura de vídeo de alta resolución temporal. Las cámaras de vídeo están limitadas por diversos factores hardware y para capturar vídeos de gran resolución espacial a altas velocidades se requiere material altamente especializado. Una alta resolución temporal es una necesidad para el estudio de ciertos fenómenos como el comportamiento de fluidos o tejidos bajo ciertas condiciones externas o la observación de fracturas de materiales.

En este proyecto se aborda el problema de la captura de vídeo de alta resolución tanto temporal como espacial desde la perspectiva de la teoría de *compressive sensing*. El objetivo es explotar su potencial para lograr reconstruir un vídeo completo a partir de una *única* imagen en la que se encuentra codificada la información temporal. Esto se consigue realizando la captura del vídeo con una exposición codificada, es decir, muestreando diferentes píxeles en cada fotograma en función del tiempo, para después recuperar toda la información necesaria para reconstruir el vídeo completo mediante procesado basado en *compressive sensing*.

Agradecimientos

Este proyecto ha sido desarrollado dentro del Graphics and Imaging Lab, perteneciente al Grupo de Informática Gráfica Avanzada (GIGA) de la Universidad de Zaragoza.

Me gustaría expresar mi gratitud hacia mi directora de proyecto, Belén Masiá por su inestimable dirección y ayuda, lo que ha hecho el desarrollo de este proyecto un proceso interesante y altamente gratificante. Me gustaría agradecer también a Diego Gutiérrez por introducirme en el mundo de la informática gráfica y por la oportunidad de trabajar en su grupo. También quiero extender mi agradecimiento a todo el GIGA por su acogida y ayuda.

Por otra parte, me gustaría agradecer al departamento de informática de la universidad Rey Juan Carlos de Madrid por aportar algunos de los vídeos utilizados en este proyecto, en particular a Miguel Ángel Otaduy y David Miraut. También al departamento de *Laser & Optical Technologies* del instituto universitario de investigación en ingeniería de Aragón (I3A), en concreto a María Pilar Arroyo por facilitarme el equipamiento, y a Laura Arévalo y Eva Roche por ayudarme a capturar el resto de los vídeos utilizados.

A nivel personal, quiero agradecer su continuo apoyo durante toda la carrera a mi familia y amigos, en especial a mis padres, que siempre me han apoyado a pesar de los malos momentos, y a mi compañero Samuel Navarro ya que sin su ayuda no hubiese llegado hasta aquí.

Por último, me gustaría dar las gracias a todos los profesores que han aportado algo a mi formación a lo largo de todo el trayecto, como persona y como ingeniera.

Índice

Índice de figuras

1. Introducción

1.1. Objetivo, alcance y contexto del proyecto

En los últimos años, se ha acelerado el avance en tecnologías de captura de vídeo, propiciado por la necesidad de capturar información a altas resoluciones tanto temporales como espaciales. Sin embargo, las cámaras todavía se enfrentan a un compromiso entre resolución espacial y resolución temporal a la hora de realizar el proceso de captura de vídeo. Esto quiere decir que, durante la captura, un aumento de la resolución temporal (número de imágenes por segundo) se logra disminuyendo la resolución espacial (número de píxeles capturados) y viceversa. Este compromiso se presenta debido a limitaciones en el hardware de los dispositivos de captura, como por ejemplo el tiempo de lectura y conversión analógico-digital de los sensores de imagen. Actualmente, es posible acelerar este proceso introduciendo conversores A/D en paralelo y buffers de imagen, como demuestran Kleinfelder et al. [6], pero esto generalmente requiere más transistores por píxel y, por tanto, aumenta el coste y el tamaño de los dispositivos.

A pesar de la continua mejora de los dispositivos de captura de vídeo, este compromiso entre resolución espacial y temporal se sigue manteniendo. Ciertas aplicaciones, como la visualización del comportamiento de fluidos o el estudio de la fractura de materiales, requieren altos valores de resolución temporal y espacial al mismo tiempo, lo cual actualmente requiere contar con material altamente especializado y costoso. Este proyecto trata sobre una alternativa para la obtención de vídeos de alta resolución espacio-temporal que hace uso de la teoría de *compressive sensing*.

Compressive sensing es un campo de gran expansión y rápido desarrollo en los últimos años, y en el que muchos autores han trabajado debido a su enorme potencial. Convencionalmente, a la hora de capturar un vídeo (o cualquier otra señal), el muestreo está limitado por el teorema de Nyquist-Shannon, que dicta que para reconstruir perfectamente una señal, ésta se debe muestrear al menos al doble de la frecuencia máxima de la misma. Esto, sumado a las limitaciones físicas del dispositivo de captura, supone una restricción en la captura de vídeo. Es por ésto que *compressive sensing* es de tanto interés, porque permite la reconstrucción de señales incluso aunque no se cumpla este teorema. *Compressive sensing* se ha aplicado en múltiples dominios y áreas de conocimiento, pero un ejemplo de su enorme potencial en el ámbito de la captura de imagen es la *single pixel camera* de Wakin et al. [1], una cámara de un solo píxel que es capaz de reconstruir imágenes completas a partir de un cierto número de medidas, mucho menor del número de píxeles de la imagen obtenida. La arquitectura de esta cámara puede verse en la Figura 1.1.

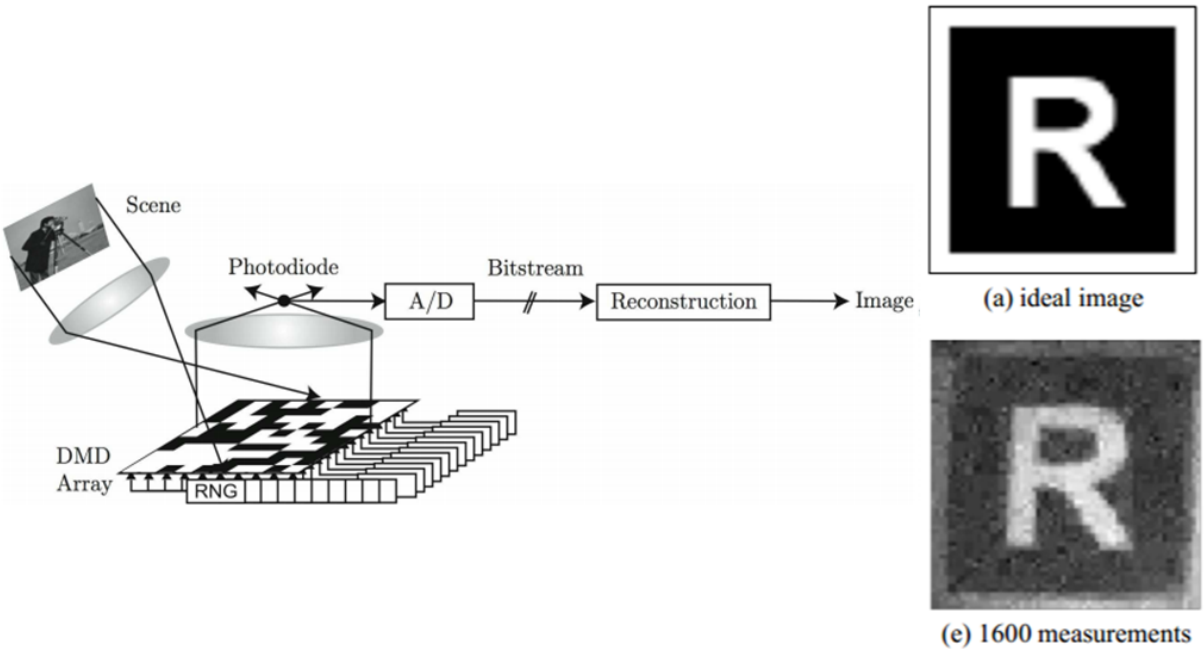

Figura 1.1: Diagrama de bloques de la *single pixel camera*. La luz incidente de la escena es reflejada en un dispositivo digital de micro-espejos (DMD) con un patrón aleatorio que dirige solo ciertos rayos de luz hacia el fotodiodo que conforma el sensor. Cada patrón diferente produce un voltaje en el fotodiodo que corresponde con una medida. Usando técnicas de *compressive sensing* es posible reconstruir la imagen original a partir de un cierto número de medidas [1].

Para que la reconstrucción a partir de las muestras capturadas se pueda llevar a cabo se necesitan cumplir una serie de condiciones y contar con una serie de elementos. Primero, se necesita encontrar una base en la que la señal sea *sparse*, esto es, que pueda representarse en el dominio de dicha base con una cantidad baja de coeficientes. Esto puede parecer complicado pero puede lograrse con el uso de algoritmos de entrenamiento de diccionarios a partir de bases de datos de señales del mismo tipo que las que se quiere recuperar, o buscando una base adecuada entre la gran cantidad de ya existentes. Segundo, el patrón con el que se muestrea la señal (la matriz de medida) debe cumplir una serie de condiciones aunque, como vamos a ver más adelante, un patrón de muestreo aleatorio cumple con alta probabilidad estas condiciones. Finalmente, se necesita un algoritmo de reconstrucción que sea capaz de hallar los coeficientes de la representación de la señal en la base. Este esquema de reconstrucción se presenta de manera básica en la Figura 1.2.

Una manera sencilla de entender *compressive sensing* es compararlo con un proceso conocido de compresión. El estándar de compresión JPEG [7] hace uso de la transformada del coseno directa (DCT) para comprimir imágenes. El proceso simplificado consiste en, una vez se tiene la imagen, calcular sus coeficientes de representación en el dominio de la DCT y después, descartar los coeficientes más pequeños, es decir, los que contienen menor cantidad de información. En el caso de *compressive sensing*, se trata de realizar este *descarte* directamente durante el proceso de adquisición, es decir, elimina la necesidad de capturar la imagen completa para después descartar información. La dificultad radica en que, mientras que en un proceso de compresión inicialmente se tiene toda la señal para poder elegir qué se quiere descartar, en *compressive sensing* no se cuenta con la señal completa.

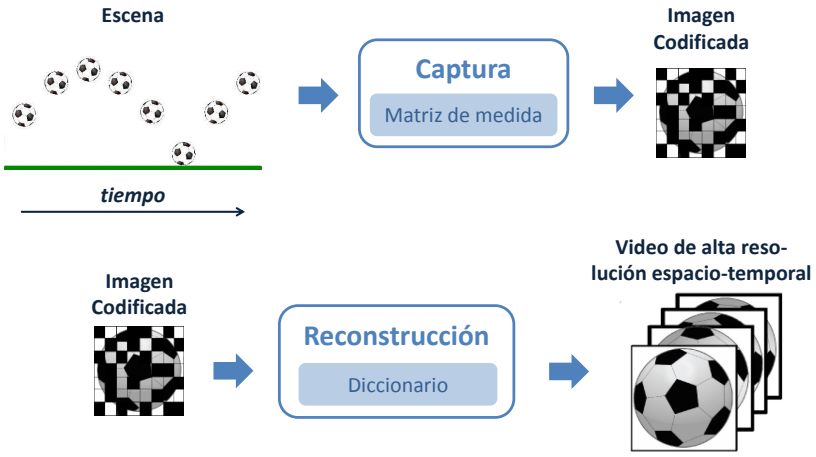

Figura 1.2: Esquema básico de captura de vídeo mediante *compressive sensing*.

El objetivo de este proyecto es explotar el potencial de *compressive sensing* para lograr la reconstrucción de un vídeo a partir de una sola imagen capturada utilizando un cierto patrón de muestreo o matriz de medida, basándose en un sistema presentado por Hitomi et al. [8, 9]. La meta final es lograr un sistema realizable físicamente que sea capaz de capturar un evento en una sola imagen codificada y después, a partir de esa imagen, reconstruir un vídeo completo. Además, se analizarán los parámetros de dicho sistema para obtener los valores más adecuados al problema de captura de vídeo. El alcance de este proyecto incluye:

- Implementación de un sistema de obtención de vídeo mediante *compressive sensing*.
- Captura de una base de datos de vídeos de alta resolución temporal y espacial para el entrenamiento de diccionarios y el posterior análisis.
- Análisis de parámetros del sistema implementado.
- Elección de los valores de parámetros más beneficiosos y propuestas de mejora al sistema implementado en los ámbitos de entrenamiento y reconstrucción.

1.2. Organización del proyecto

La memoria parte de los conceptos básicos y generales de *compressive sensing* para después aplicarlos al caso concreto de la captura de vídeo. En el Capítulo 2 se presenta este proyecto en el marco contextual, relacionándolo con otros trabajos a lo largo de estos últimos años. En el Capítulo 3 se introduce el marco teórico general, en el que se explican las bases y los elementos necesarios para un sistema de *compressive sensing*, para después, en el Capítulo 4 extender estos conceptos al caso concreto de captura de vídeo, describiendo el sistema implementado en este proyecto. En el Capítulo 5 se hace una descripción del proceso de captura de la base de datos. Después, en el Capítulo 6, se realiza una caracterización de los vídeos capturados, así como un análisis del sistema explorando diferentes parámetros y proponiendo ciertas mejoras. Posteriormente, en el capítulo 7, se presentan los resultados, y en el Capítulo 8 las conclusiones del proyecto así como posible trabajo futuro.

1.3. Planificación

Durante los siete meses aproximados de duración del proyecto, el trabajo se ha dividido en las siguientes tareas: documentación del marco teórico de *compressive sensing*, documentación de *compressive sensing* aplicado a reconstrucción de imágenes y algoritmos de reconstrucción, implementación de un sistema de reconstrucción de imágenes basado en *compressive sensing*, documentación e incorporación del entrenamiento de diccionarios al sistema 2D , documentación de *compressive sensing* aplicado a captura de vídeo, extensión del sistema 2D a un sistema de captura y reconstrucción de vídeo, captura de la base de datos utilizada en las simulaciones, familiarización con el uso del clúster *HERMES*, diseño de los experimentos para los diferentes parámetros, entrenamiento de los diccionarios para la reconstrucción de vídeo y experimentos con diferentes parámetros de la reconstrucción, análisis de resultados, y redacción de la memoria. La distribución de estas tareas a lo largo del tiempo se pueden ver en la Figura 1.3. Todas las tareas de implementación y simulación han sido llevadas a cabo con el software Matlab y para el cálculo de experimentos computacionalmente intensivos se ha utilizado el cúster *HERMES* del i3A.

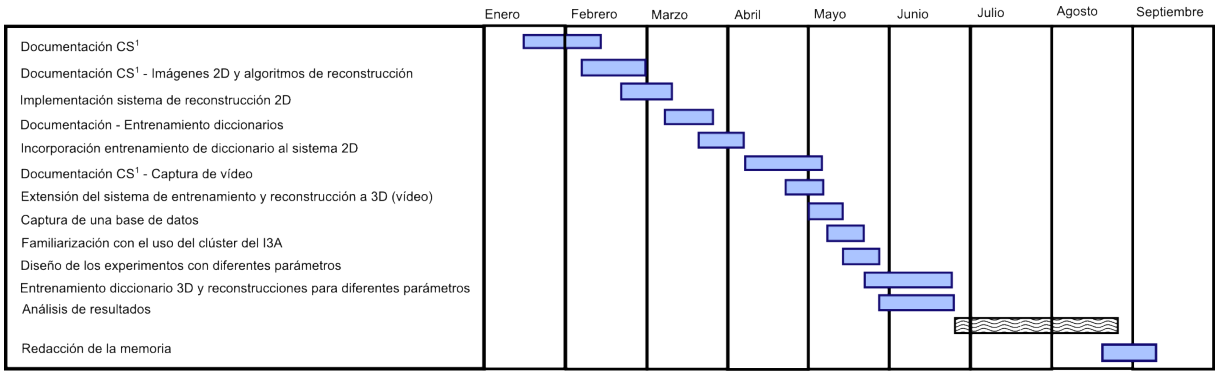

Figura 1.3: Diagrama de Gantt de las actividades realizadas a lo largo del proyecto.

2. Trabajo relacionado

Está comúnmente aceptado que el área de *compressive sensing* surge a raíz de la publicación, en 2006, del trabajo seminal de Candes [10], en el que se demuestra matemáticamente que dada una señal dispersa (*sparse* según el término en inglés que se usará de ahora en adelante) en algún dominio, es posible recuperar la señal original a partir de un número de muestras menor que el dictado por el teorema de Nyquist-Shannon. La publicación de este trabajo dio lugar, en un corto espacio de tiempo, a un gran número de trabajos que profundizaron en el desarrollo del marco matemático en el que se apoya este campo. Ejemplos de esto son el trabajo de DeVore [11], que analiza los resultados de muestrear de manera determinista, o el de Rauhut [12] que extiende *compressive sensing* a señales que no son *sparse* en una base ortonormal pero sí en un diccionario redundante.

Una vez sentadas las bases, un gran número de autores profundizaron en posibles algoritmos de reconstrucción de la señal a partir del conjunto reducido de muestras, como por ejemplo el trabajo de Wang [13] sobre *Orthogonal Matching Pursuit* para este tipo de problema, o el de Temlyakov [14] donde habla sobre la eficiencia de los algoritmos voraces. Otra tendencia se dedicó a la exploración de bases y diccionarios para diferentes aplicaciones, como Gurevich y Hadani [15] que se centran en la búsqueda de diccionarios que cumplan ciertas condiciones necesarias para su uso en *compressive sensing*, o la búsqueda de patrones de muestreo óptimos como Sebert [16], que propone el uso de matrices Toeplitz como matrices de muestreo.

Tras estos trabajos genéricos, no aplicados a un dominio concreto, empezaron a surgir trabajos que aplicaban la teoría de *compressive sensing* a un dominio concreto. Ejemplos de esto son diversos trabajos en el campo de la imagen médica, como Provost y Lesage [17] que aplican *compressive sensing* para la captura de tomografías fotoacústicas o Lustig et al. [18] para la captura de resonancias magnéticas. En el campo de las comunicaciones, algunos ejemplos son Bajwa [19] para la estimación eficiente de canales *sparse* multi-antena, o Bajwa et al. [20] para la estimación de canales multi-camino.

Otro de los campos en los que *compressive sensing* ha sido ampliamente utilizado es el campo de imagen y vídeo. Uno de los ejemplos más significativos es la *single pixel camera* de Wakin et al. [1], donde proponen un prototipo de cámara de un solo píxel capaz de reconstruir imágenes completas mediante *compressive sensing* y varias exposiciones. Otro ejemplo es el trabajo de Marwah et al. [21], en el que logran la adquisición de *light fields* (estructuras tetradimensionales que contienen información espacial y angular de una escena) a partir de una sola imagen codificada, o un reciente trabajo en el que se obtienen imágenes hiperespectrales de alta resolución espacial, de nuevo a partir de una sola imagen codificada y mediante *compressive sensing* [22]. Finalmente, el trabajo de Ito et al. [23] utiliza compressive sensing para permitir, a partir de un

número muy reducido de imágenes de una escena, reproducir todo tipo de efectos fotográficos (como variación del enfoque o de la apertura) en post-proceso.

Por otra parte, el aprendizaje de diccionarios especialmente aplicados al caso de imagen y vídeo ha sido estudiado, entre otros, por Aharon et al. [5] donde proponen el algoritmo K-SVD como método de aprendizaje de diccionarios y prueban sus resultados en reconstrucciones mediante *compressive sensing* de imágenes 2D sub-muestreadas.

En el caso de captura de vídeo, se ha intentado anteriormente la captura de vídeo de alta resolución espacial y temporal mediante otros enfoques no basados en *compressive sensing*. Gupta et al. [24] proponen un método para recuperar vídeos de alta resolución a partir de vídeos en baja resolución y unos pocos fotogramas clave en alta resolución. Wilburn et al. en su trabajo [25] proponen la captura de vídeos de alta velocidad mediante el prototipo formado por una matriz de cámaras mostrado en la Figura 2.1.

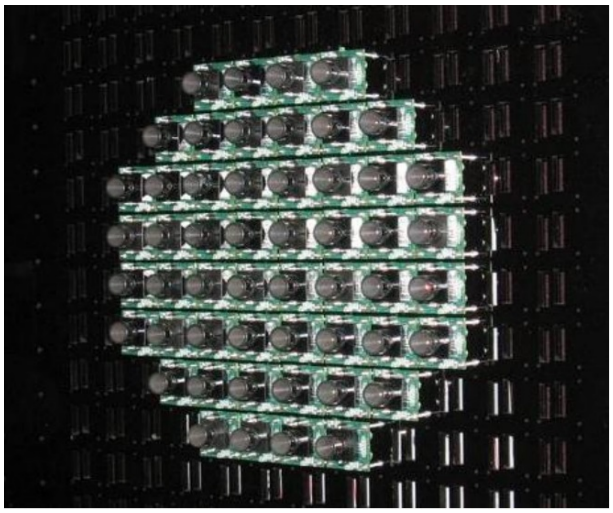

Figura 2.1: Matriz de 52 cámaras usado por Wilburn et al. para capturar vídeo de alta velocidad [25].

Además, el uso de exposiciones codificadas, i.e patrones de muestreo a la hora de capturar imagen o vídeo, no ha sido solo utilizado en *compressive sensing*, sino que también ha sido ampliamente utilizado anteriormente como método para mejorar algunos aspectos en la captura de imagen o vídeo en el campo de la fotografía computacional. El objetivo del uso de exposiciones codificadas es codificar la señal antes de que llegue al sensor, bien sea en el espacio, en forma de aperturas codificadas, o en el tiempo, en forma de funciones de obturación. El caso de funciones de obturación (*shutter*) que varían en el tiempo es de especial interés por su relación con este proyecto y la captura de vídeo. Entre ellas, en [26], Raskar et al. proponen el uso de la función llamada *flutter shutter*, que puede verse en la Figura 2.2, para eliminar *motion-blur* en captura de imágenes. Con el mismo propósito, en [2] se propone una alternativa al convencional *rolling shutter*, llamada *coded rolling shutter* cuyo diagrama puede verse también en la Figura 2.2. Por otra parte, el uso de aperturas codificadas para codificar la imagen especialmente antes de que llegue al sensor, ha sido estudiado, por ejemplo, por Masia et al. [3], que utilizan aperturas codificadas como la mostrada en la Figura 2.3 para eliminar el desenfoque en la captura de imágenes.

Finalmente, es de especial interés el trabajo de Hitomi et al. [8, 9], donde se proponen dos

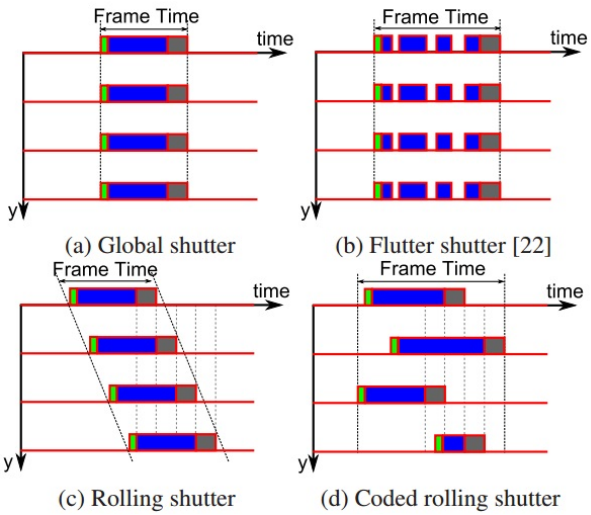

Figura 2.2: Diagrama temporal para cuatro tipos de *shutter*. (a) *Shutter* global: Muestreo continuo en un intervalo de tiempo, (b) *flutter shutter*: muestreo no continuo en intervalos aleatorios, (c) *rolling shutter*: muestreo continuo de diferentes intervalos de tiempo a lo largo de las filas de la imagen, (d) *coded rolling shutter*: Modificación de la secuencia de muestreo del *rolling shutter* convencional mediante una función de optimización [2].

Figura 2.3: Izquierda: Cámara desmontada usada en los tests de Masia et al. para corrección de *blur* por desenfoque. Derecha: Lente con una de las aperturas codificadas insertada [3].

puntos clave para este proyecto. Por una parte, proponen la arquitectura de un sistema capaz de recuperar un fragmento de vídeo de una sola imagen con exposición codificada mediante aprendizaje de diccionarios. Los resultados en simulación se contrastan con resultados prácticos mediante el uso de un prototipo creado con un dispositivo LCoS (cristal líquido sobre silicio). Por otra parte, presentan una nueva codificación para el obturador basado en un muestreo aleatorio de píxeles para cada fotograma y prueban sus resultados en la reconstrucción de vídeo a partir de una sola imagen tal y como se muestra más adelante en este proyecto.

3. Compressive sensing: marco teórico

En este capítulo se presentan las bases teóricas genéricas de *compressive sensing* necesarias para entender cómo se puede reconstruir un vídeo completo de una sola imagen. Primeramente se presentan las ecuaciones básicas para después concluir con un ejemplo sencillo de aplicación.

Sea una señal de interés X de valores reales, unidimensional y discreta de duración N. Dicha señal puede representarse como un vector columna $N \times 1$ en R^N con elementos $X[n]$, con $n = 1, 2, \ldots, N$.

Sea una base ortonormal $\{\psi_i\}_{i=1}^{K}$ representada matricialmente de manera que $\psi = [\psi_1|\psi_2|\ldots|\psi_K]$ con cada vector de la base ψ_i como una columna de la matriz.

La señal X puede expresarse como una combinación lineal de los vectores de la base ψ

$$X = \sum_{i=1}^{K} \alpha_i \psi i \tag{3.1}$$

donde α son los coeficientes de dicha combinación $\alpha_i = \langle X, \psi_i \rangle = \psi_i^T X$

También puede representarse la Ecuación 3.1 de manera matricial, que es como se utilizará de aquí en adelante, tal y como se presenta en la siguiente ecuación

$$X = \psi\alpha \tag{3.2}$$

Se dice que una señal es *k-sparse* cuando puede expresarse como combinación lineal de k vectores de la base ψ, es decir $(K - k)$ elementos del vector α son cero. Cuando $k << K$, es decir, la información se encuentra contenida tan solo en unos pocos coeficientes, se dice que la señal es compresible.

Convencionalmente, el proceso de compresión se realiza capturando la señal completa para posteriormente transformarla en alguna base en la que sea compresible y descartar los coeficientes menos significativos (por ejemplo DCT para el estándar JPEG). El objetivo de *compressive sensing* es incrementar la eficiencia de este proceso de manera que se logre capturar la señal ya comprimida, es decir, realizar ese descarte de información en el proceso de muestreo en lugar de posteriormente en el procesado.

A este efecto, se define la *matriz de medida* ϕ, que es el concepto matemático que precede a la realización física que sería el obturador. Es la colección de vectores utilizados para aplicar un

proceso de muestreo lineal que computa $M < N$ productos internos entre X y la colección de vectores $\{\phi_j\}_{j=1}^{M}$ de manera que

$$Y_j = \langle X, \phi_j \rangle \tag{3.3}$$

Si se expresa $Y = [Y_1 \ \ldots \ Y_M]^T$ como un vector columna de $M \times 1$ elementos y se organizan los elementos de la base ϕ como filas en una matriz $M \times N$ y usando la Ecuación 3.2 se puede representar el proceso completo de *compressive sensing* de acuerdo a la Figura 3.1 y la siguiente ecuación:

$$Y = \phi X = \phi\psi\alpha = \Theta\alpha \tag{3.4}$$

donde $\Theta = \phi\psi$ es una matriz de tamaño $M \times K$ que representa conjuntamente la base y la matriz de medida.

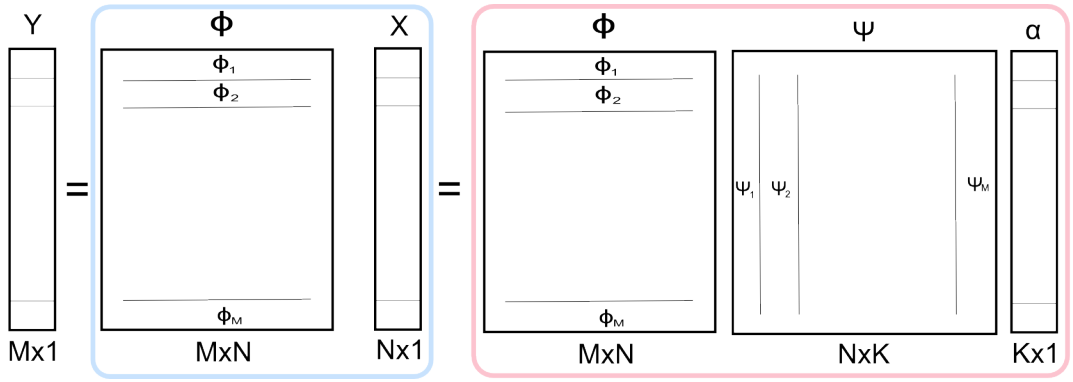

Figura 3.1: Representación del proceso de *compressive sensing* acuerdo a la ecuación 3.4. La señal Y de tamaño $M \times 1$ procede de la señal X de tamaño $N \times 1$ con $N > M$ muestreada con la matriz de medida ϕ. Finalmente, la señal X puede expresarse también mediante la base ψ de K elementos y el vector de coeficientes de representación α de la señal en la base.

El proceso de muestreo no es adaptativo, esto es, la matriz de medida es siempre la misma independientemente de la señal, por lo que se presenta el problema de diseñar una matriz de medida ϕ estable capaz de muestrear correctamente toda señal compresible en la base ψ. Para que la reconstrucción sea posible la matriz de medida debe cumplir dos condiciones, la primera es cumplir la propiedad de isometría restringida (RIP) y la segunda es que dicha matriz sea incoherente con la base ψ. Una descripción en detalle de estas propiedades se puede encontrar en el Anexo A.

El diseño de una matriz de medida que cumpla estas dos propiedades y que sea realizable desde un punto de vista práctico es un proceso complejo. Sin embargo, una manera sencilla de cumplir satisfactoriamente las dos condiciones con alta probabilidad, es construir ϕ como una matriz aleatoria como demuestran Baraniuk et al. [27].

Finalmente, utilizando la Ecuación 3.4 y conociendo la matriz de medida ϕ y la señal muestreada Y, es posible reconstruir la señal original X mediante un problema de minimización que se planteará en la Sección 4.4 del Capítulo 4.

Los factores principales que deben considerarse a la hora de diseñar un sistema en el marco teórico de *compressive sensing*, y que se analizarán en el siguiente capítulo son: Selección de una base en la que el conjunto de señales de interés sea *sparse*, diseño de una matriz de medida que

permita recuperar la señal original evitando pérdidas de información, y elección de un algoritmo adecuado que sea capaz de calcular los coeficientes α para recuperar dicha señal.

A continuación se presenta un ejemplo sencillo propuesto por Romberg en [4] de aplicación de *compressive sensing* a una señal unidimensional para ilustrar lo expuesto anteriormente.

3.1. Ejemplo de aplicación: Señales unidimensionales

Sea una señal discreta de 256 muestras compuesta por 16 sinusoides complejas de frecuencia desconocida como puede verse en la Figura 3.2. Se puede decir que esta señal es *sparse* en el dominio de Fourier ya que la mayor parte de los coeficientes son cero.

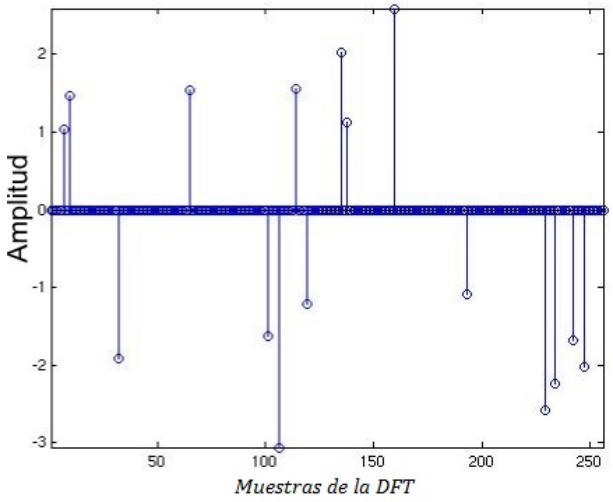

Figura 3.2: Transformada de Fourier de una señal discreta unidimensional de 256 muestras [4].

Dado que no está limitada en banda, para una reconstrucción completa de acuerdo al teorema de Nyquist, en el dominio del tiempo se deben muestrear las 256 componentes. Ahora, suponer que se quiere aplicar la teoría de *compressive sensing* y se muestrean aleatoriamente solo 80 muestras en el tiempo de acuerdo a la Figura 3.3.

Se tiene pues, la señal original de 256 muestras, y la versión muestreada de la misma con tan solo 80 muestras. El conjunto de señales de longitud 256 muestras que podrían dar lugar a una misma señal sub-muestreada de 80 muestras es un subespacio de dimensión $256 - 80 = 176$. Sin embargo, dado que se sabe que la señal es *sparse* en el dominio de Fourier, la solución, como se verá más adelante, será aquella señal cuya DFT tenga norma L1 mínima, es decir, que la suma de los coeficientes de la transformada de Fourier sea la menor posible o lo que es lo mismo, que la mayor parte de sus coeficientes sean cero, dando lugar con ello a una señal *sparse* en la reconstrucción. De esta manera, es posible extender este mismo planteamiento a señales de todo tipo: 2D (imagen), 3D (vídeo), 4D (lightfields), etc. Por otra parte, en este ejemplo sencillo es posible recuperar la señal perfectamente. Sin embargo, como se verá más adelante, esto no siempre es posible en la práctica debido a diferentes factores; por ejemplo, señales que en la práctica no serán totalmente *sparse*, es decir, que la mayor parte de sus coeficientes serán muy

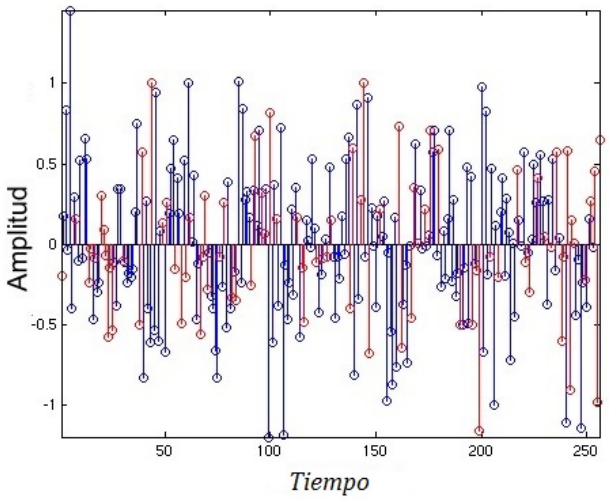

Figura 3.3: Señal discreta unidimensional en el dominio temporal. Muestras originales de la señal en azul y muestreadas en rojo [4].

bajos pero no estrictamente cero, o matrices de medida que no serán totalmente aleatorias.

4. Captura de vídeo mediante compressive sensing

4.1. Introducción

En este capítulo se introducen los conceptos y parámetros presentes en un sistema de captura y reconstrucción de vídeo mediante *compressive sensing* para más tarde en el Capítulo 7 analizar su repercusión en los resultados de una reconstrucción.

La teoría de *compressive sensing* puede aplicarse a todo proceso que requiera un muestreo siempre y cuando se cumplan las condiciones para permitir la reconstrucción de la señal, es decir, siempre que la señal sea *sparse* en un dominio y se cuente con una matriz de medida adecuada. En este caso, el proceso que se presenta es la captura de vídeo.

En un dispositivo convencional, el proceso de captura de un vídeo se realiza mediante el muestreo todos los píxeles para cada fotograma de tamaño $M \times N$ píxeles y para T instantes temporales, lo que hace un total de $M \times N \times T$ muestras. El objetivo al abordar este proceso desde la perspectiva de *compressive sensing* es lograr reconstruir dicho vídeo a partir de un número de muestras menor que el que se requiere convencionalmente, en este caso a partir de una única imagen, es decir, $M \times N$ muestras. A este efecto, tal y como se ha visto en el capítulo anterior, se puede dividir el problema en tres factores principales a considerar:

- **Base**: Conjunto de vectores linealmente independientes capaces de representar la señal, en este caso vídeo, de manera *sparse*, es decir, con un reducido número de coeficientes distintos de cero. Llamadas **diccionarios** en este campo de aplicación.
- **Matriz de medida**: Matriz que define cómo se va a muestrear la señal, en este caso mediante una exposición codificada, esto es, una función de obturación que varía los píxeles a muestrear en cada instante temporal, y que debe cumplir las condiciones presentadas en el Capítulo 3. En el caso de captura de vídeo este concepto se materializa en forma de un obturador.
- **Algoritmo de reconstrucción**: Algoritmo que sea capaz de calcular los coeficientes de representación α de la señal en determinada base ψ de manera *sparse*.

Es necesario relacionar el caso concreto de captura de vídeo con las bases teóricas de *compressive sensing* expuestas en el Capítulo 3. Para ello se define la señal original X como un bloque

de vídeo de duración T fotogramas y $M \times N$ píxeles por fotograma. La señal capturada Y es una proyección de dos dimensiones (una imagen) del espacio tridimensional de la señal original X (un vídeo) y queda definida muestreando dicha señal X con la matriz de medida ϕ, que en este caso es un obturador. El resultado es una sola imagen de tamaño $M \times N$ píxeles, obtenida como una combinación de los fotogramas del vídeo original. Un ejemplo de este proceso puede verse en la Figura 4.1, en la que se ilustra cómo un fragmento de un vídeo de una llama prendiéndose se captura en una sola imagen mediante una matriz de medida que muestrea píxeles aleatorios. La forma de esta matriz de medida, es decir, la manera en la que debe muestrearse el vídeo original, se discute en profundidad más adelante en la Sección 4.3.

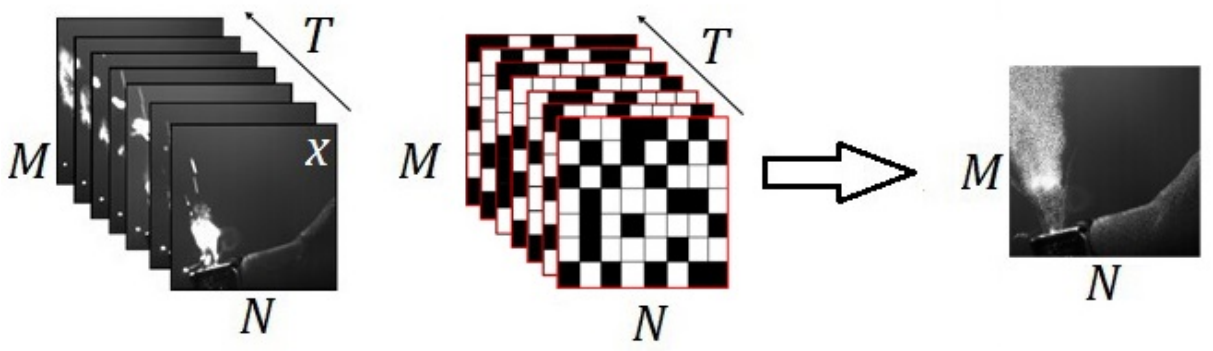

Figura 4.1: Proceso de captura de un vídeo de tamaño $M \times N$ píxeles y duración T fotogramas mediante una matriz de medida ϕ que muestrea píxeles aleatorios para cada fotograma del vídeo con el objetivo de construir la imagen Y de tamaño $M \times N$ píxeles.

Una vez obtenida la captura de vídeo en forma de una única imagen, se plantea la problemática de la reconstrucción en la Sección 4.4, así como los posibles algoritmos que pueden utilizarse.

Es muy importante remarcar que después de la captura, durante todo procesado, no se trabaja con la información (tanto vídeos como imágenes) del tamaño original sino que se divide en bloques más pequeños, de orden de la decena de píxeles. Cada uno de estos bloques más pequeños se trata por separado, como si de un problema independiente se tratase.

Esto es así porque a la hora de representar la información en un diccionario, nos interesa ser capaces de representar *características* clave, como bordes o ciertos movimientos, que puedan utilizarse después para conformar, poco a poco, un vídeo completo. No tendría sentido buscar un diccionario capaz de representar vídeos completos porque se necesitaría un diccionario muy específico, adaptado a dichos vídeos, lo cual restringiría mucho la generalización del diccionario para un conjunto más amplio.

Para esto, la imagen Y debe dividirse en sub-imágenes más pequeñas de tamaño $p \times q$ con $p < M$ y $q < N$. Con ello, el problema de la reconstrucción del vídeo completo queda reducido a varias reconstrucciones independientes de fragmentos de vídeo de menor tamaño $p \times q \times T$, que será el tamaño de bloque con el que se hayan entrenado los diccionarios. El esquema de todo el proceso de captura y reconstrucción de vídeo seguido durante este proyecto se muestra en la Figura 4.2.

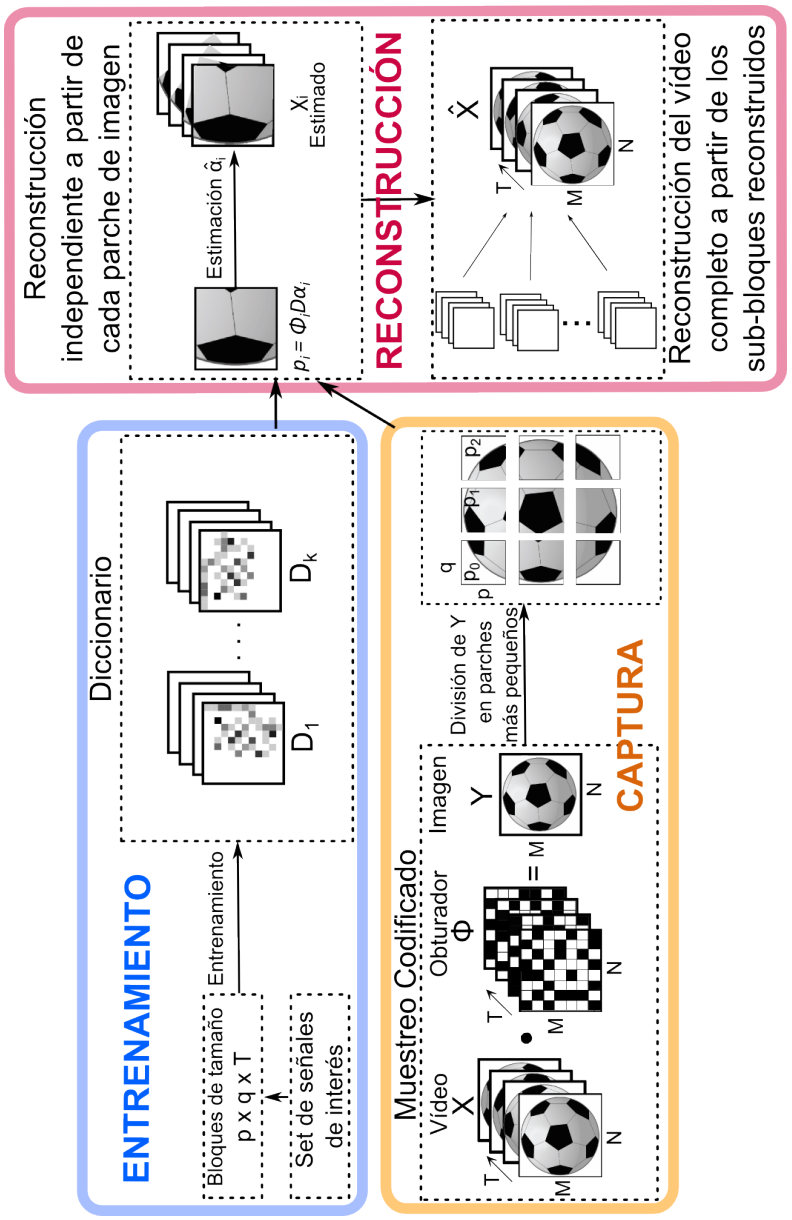

Figura 4.2: Arquitectura del sistema de captura y reconstrucción de vídeo mediante *compressive sensing*. Arriba: Entrenamiento de un diccionario de K elementos apropiado para el conjunto de señales de interés. Abajo: Muestreo de la señal original de vídeo X mediante una matriz de medida ϕ resultando en la imagen Y. División de la imagen Y en parches de menor tamaño. Derecha: Reconstrucción de sub-bloques de vídeo a partir de cada parche como un problema independiente y composición del vídeo original completo.

4.2. Diccionario

Como se ha explicado anteriormente, es necesario un sistema de representación adecuado para el caso de vídeo, es decir, un diccionario. Para ello, se puede generalizar una clasificación entre dos opciones, la alternativa no adaptativa y la adaptativa.

La opción **no adaptativa** consiste en escoger un diccionario de entre la gran cantidad de sistemas de representación ya existentes, como por ejemplo, en el caso de imagen DCTs (Discrete Cosine Transform) o Wavelets. La ventaja en este caso radica en que estos diccionarios ya están matemáticamente definidos y sus propiedades son conocidas, así como la existencia de transformadas rápidas y computacionalmente eficientes en sus respectivos dominios. Por otra parte, el uso de estos diccionarios no proporciona unos coeficientes óptimamente *sparse* para la señal de interés.

Esto puede mejorarse con la opción **adaptativa**: mediante el uso de diccionarios aprendidos con muestras de la señal de interés, es decir, adaptados a dicha señal. Para entrenar un diccionario es necesario un conjunto de señales de interés (es decir, de la misma naturaleza de las señales que se quieren representar), así como un algoritmo de entrenamiento. En este proyecto, las señales de interés son vídeos de alta resolución temporal (1000 fps - fotogramas por segundo) de todo tipo de escenarios. La descripción de la base de datos así como detalles acerca de su obtención, pueden encontrarse en el Capítulo 5, no obstante, un ejemplo de los vídeos utilizados puede verse en la Figura 4.3.

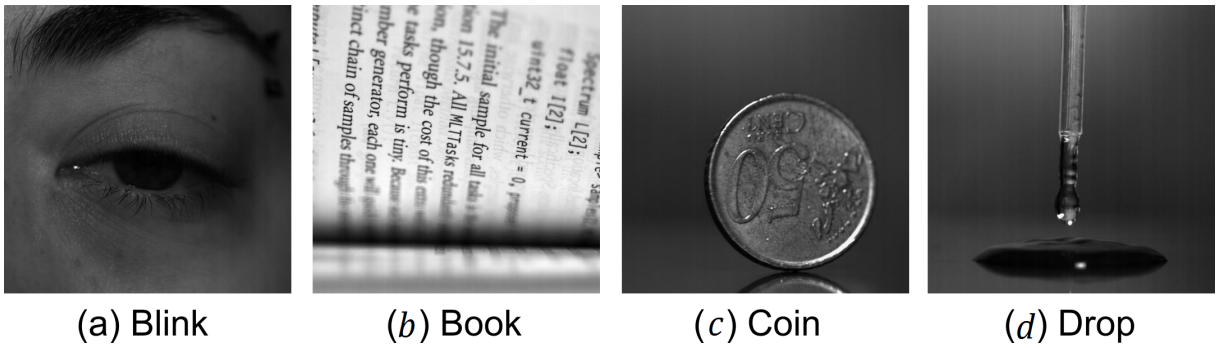

Figura 4.3: Fotogramas de algunos de los vídeos utilizados para el entrenamiento de diccionarios. (a) Parpadeo de un ojo, (b) Páginas de un libro siendo pasadas, (c) Una moneda girando, (d) Flujo de agua cayendo de un cuentagotas a una superficie plana.

A la hora del entrenamiento, uno de los algoritmos más conocidos y más usados es el algoritmo K-SVD introducido por Aharon, Elad y Bruckstein [5], motivo por el cual se usa en este proyecto. Los detalles acerca del funcionamiento de este algoritmo pueden consultarse en el Anexo B. Para representar la señal de vídeo en el dominio del diccionario de acuerdo a la Ecuación 3.2 es necesario conocer los coeficientes α. Para el cálculo de dichos coeficientes se utiliza el mismo algoritmo que en la reconstrucción, que será discutido más adelante en la Sección 4.4.

Finalmente, como resultado del entrenamiento, se obtiene un diccionario de K elementos, siendo el número de elementos K mayor o igual que el tamaño de la señal a representar, en este caso $p \times q \times T$. Cada uno de los elementos de este diccionario es llamado átomo. En el caso de vídeo, cada átomo es un bloque de vídeo de tamaño $p \times q \times T$. Un ejemplo de un átomo de uno

de los diccionarios entrenados puede verse en la Figura 4.4.

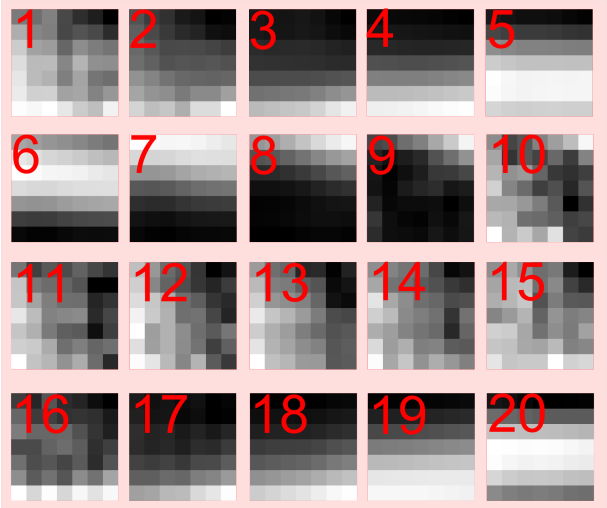

Figura 4.4: Fotogramas procedentes de un átomo aleatorio de un diccionario entrenado con bloques de vídeo de tamaño $p \times q$ con $p = 7$ y $q = 7$ y duración $T = 20 fotogramas$. Los fotogramas están ordenados temporalmente de izquierda a derecha y de arriba a abajo. En este átomo se ha capturado el movimiento de un objeto blanco de abajo a arriba que puede apreciarse en los fotogramas 3 al 9 y nuevamente el mismo movimiento empezando en el fotograma 17.

El tamaño de estos átomos y las señales utilizadas para su entrenamiento son factores que influyen en la capacidad del diccionario para representar correctamente el conjunto de datos de interés y, por tanto, en los resultados de las reconstrucciones, por lo que en el Capítulo 7 se realiza un análisis de estas variables.

4.3. Matriz de medida

La matriz de medida tal y como se ha presentado en el Capítulo 3 para el caso de captura de vídeo consiste en una exposición codificada. En la práctica, esta matriz de medida se construye como un obturador. En esta sección se presentan primero las bases teóricas para después continuar en el Apartado 4.3.1 con las limitaciones en la realización física, y finalmente exponer algunos ejemplos de obturadores en el Apartado 4.3.2.

En términos de simulación, la matriz de medida es un bloque de vídeo del mismo tamaño que la señal a capturar $M \times N \times T$ con unos en los píxeles que van a ser capturados y ceros en los píxeles que no. Así pues, para cada fotograma, se obtiene el producto de la señal por la máscara definida por la matriz de medida para posteriormente, sumar todos los resultados obteniendo una sola imagen combinación de todos los frames del vídeo de acuerdo a la siguiente ecuación:

$$I(x, y) = \sum_{t=1}^{T} S(x, y, t) X(x, y, t) \tag{4.1}$$

siendo $I(x, y)$ la imagen resultante del proceso, S la función del obturador y X la señal original de vídeo. En un sistema de captura convencional $S(x, y, t) = 1, \forall x, y, t$. El objetivo es lograr una

función S que cumpla las condiciones de matriz de medida lo mejor posible y que, a su vez, sea realizable en términos de hardware. Varios ejemplos de obturadores pueden verse en la Figura 4.6.

4.3.1. Limitaciones hardware

A pesar de que es posible diseñar una matriz de medida ideal, a la hora de la realización práctica, este diseño tiene que ajustarse a las limitaciones impuestas por el hardware, es decir, tiene que degradarse, en el sentido de que el obturador construido no será tan óptimo con la matriz de medida originalmente diseñada.

Los sensores de imagen se basan en tecnología CMOS (semiconductor complementario de óxido metálico), por tanto es necesario que el obturador cumpla las restricciones impuestas por esta tecnología.

Un proceso completo de captura comprende varios procesos que requieren un tiempo. Primero, el **tiempo de integración**, que es el tiempo que el obturador permanece abierto y durante el cual los píxeles están acumulando luz. Aquí aparece la primera limitación ya que el tiempo máximo de integración está limitado por el ruido térmico del sensor. A este proceso deben sumarse los tiempos de conversión analógico-digital y el tiempo de escritura, lo que resulta en una limitación en la velocidad en la que el obturador puede abrirse y cerrarse. Debido a esto y al hecho de que la mayoría de sensores CMOS no tienen un buffer incorporado en el chip, en el diseño del obturador cada píxel se ve limitado a una única exposición continua y menor o igual que el tiempo de integración de la cámara tal y como se representa en la Figura 4.5.

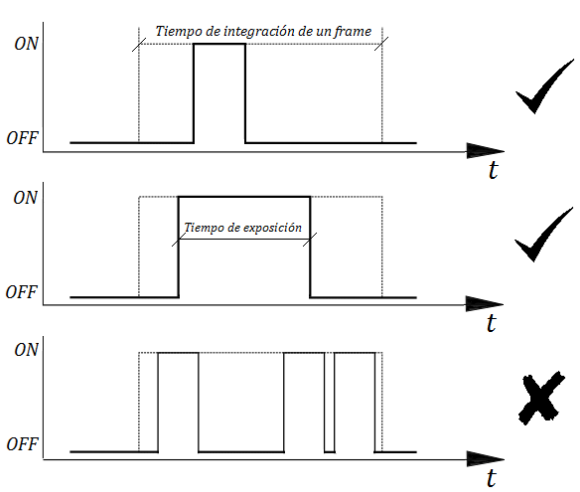

Figura 4.5: Limitaciones hardware de la matriz de medida. La exposición está limitada a un solo pulso debido a la arquitectura del sistema y la duración del pulso debe ser igual o menor al tiempo máximo de integración del sensor, condiciones que se cumplen en los dos primeros casos pero no en el tercero [8].

Esto limita el obturador a un solo pulso continuo de integración para cada píxel durante un número determinado de fotogramas, condición que se ha de tener en cuenta a la hora del diseño de la matriz de medida.

4.3.2. Funciones del obturador

Teniendo en cuenta las limitaciones explicadas en el apartado 4.3.1, la función del obturador $S(x, y, z)$ debe cumplir las siguientes condiciones:

- **Función binaria**: para un instante t, un píxel determinado puede estar muestreando (1) o no (0). Esto es $S(x, y, t) \in \{0, 1\}$.
- **Exposición en un único pulso**: debido a las limitaciones de los sensores CMOS comentadas anteriormente, cada píxel debe tener únicamente un tiempo de exposición continuo, es decir, un sólo pulso en "on"durante el tiempo de integración de la cámara.
- **Tiempo de pulso fijado para todos los píxeles**: todos los píxeles deben tener la misma duración de pulso de muestreo debido a las limitaciones prácticas de los sensores.

Existen diversas posibilidades para una función de obturador que cumpla estos requisitos. Como se ha explicado anteriormente, estos obturadores son una implementación hardware con unas limitaciones, lo cual supone que que las condiciones teóricas de una matriz de medida (RIP e incoherencia) no van a cumplirse. La calidad en las reconstrucciones con cada uno de los obturadores dependerá de cuanto se acerquen a dichas condiciones.

En este proyecto se analizan los resultados para los obturadores por Hitomi et al. [8] que, además, son los más comunes. Pueden verse en la Figura 4.6, junto con la imagen resultante del muestreo con cada uno de ellos.

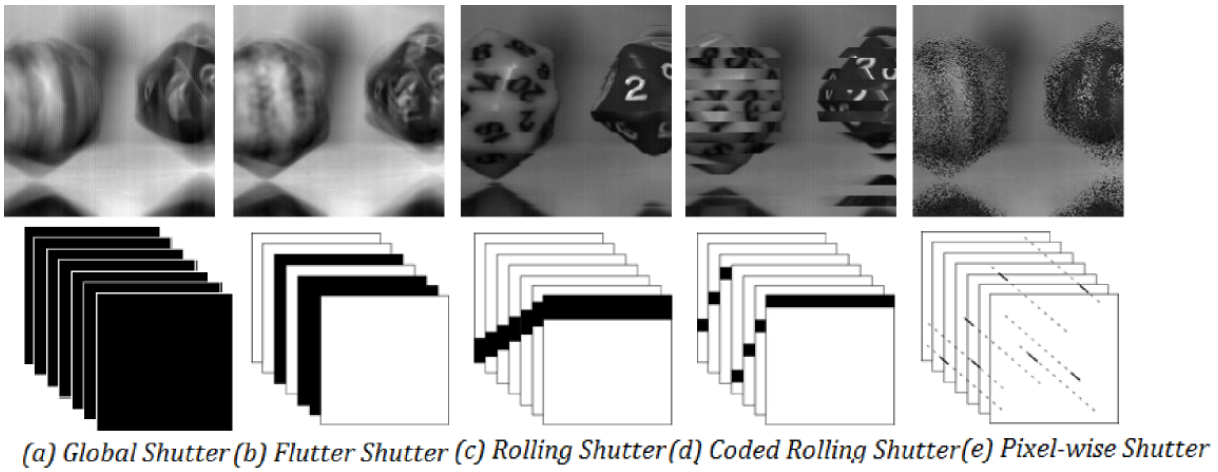

Figura 4.6: Abajo: funciones del obturador para el muestreo de vídeo. Los píxeles en negro se corresponden con píxeles siendo muestreados (on) mientras que los píxeles en blanco se corresponden con píxeles en off. Arriba: imágenes resultantes en simulación. Las imagenes correspondientes a *global shutter* y *flutter shutter* tienen más luz aparentemente debido a que en el proceso de simulación se suman todos los fotogramas (varios pulsos de integración, es decir, más luz capturada) mientras que en un proceso real se dividiría un mismo tiempo de integración en varios pulsos y, por tanto, se conservaría el mismo nivel de luz para todos las funciones.

- ***Global shutter***: Todos los píxeles se muestrean durante el tiempo de integración de la cámara, es decir, el obturador se mantiene abierto.

- ***Flutter shutter***: Se abre y se cierra el obturador de manera binaria en una secuencia pseudo-aleatoria durante el tiempo de integración de la cámara.

- ***Rolling shutter***: Se muestrean los píxeles secuencialmente por filas a lo largo del tiempo. Este obturador es comúnmente utilizado a pesar de que ocasiona una distorsión en la imagen para objetos en movimiento, ya que diferentes partes de la imagen se muestrean en tiempos diferentes.

- ***Coded-rolling shutter***: Similar al *rolling shutter*. En este caso se divide la imagen en varias sub-imágenes y se aplica *rolling shutter* a cada una de ellas por separado. Por ejemplo, en el caso de dos sub-imágenes, primero se muestrean las lineas impares aplicando *rolling shutter* y posteriormente las pares de la misma manera.

- ***Pixel-wise shutter***: Cada píxel se muestrea durante un tiempo de pulso fijado (mismo tiempo de pulso para todos los píxeles) de la manera más aleatoria posible. No obstante, para conseguir tener suficientes muestras para la reconstrucción del vídeo se ha de imponer una condición: teniendo en cuenta que la imagen resultante del proceso se va a subdividir en parches solapados para su recuperación tal y como se muestra en la Figura 4.8, para cada parche y para todo instante temporal se debe tener al menos un píxel muestreado en cada uno de esos parches, esto es $\sum_{(x,y)\in p_j} S(x,y,t) \geq 1$ para $t = 1 \dots T$ con T el número total de fotogramas a muestrear.

4.4. Reconstrucción

El objetivo de esta etapa es reconstruir el evento original a partir de una única imagen Y capturada mediante el obturador elegido. Tras dividir la imagen en parches de imagen de menor tamaño, se tienen tantas reconstrucciones independientes a resolver como parches en los que se ha dividido la imagen. A partir de cada parche de imagen de tamaño $p \times q$, de un diccionario con átomos de tamaño $p \times q \times T$ y conociendo el obturador utilizado para el muestreo, es posible recuperar un fragmento de vídeo de tamaño $p \times q \times T$. Para ello, primero se deben estimar los coeficientes α de la representación de dicho vídeo en el dominio del diccionario a partir de la imagen capturada resolviendo la Ecuación 3.4. Después, conocido el vector de coeficientes α, se puede representar el vídeo mediante la Ecuación 3.2 y obtener así la reconstrucción. Puesto que lo que se busca es una representación *sparse*, la problemática presentada se plantea como un problema de optimización de la siguiente manera:

$$\min_{\alpha} \|\alpha\|_0 \text{ sujeto a } Y = \Theta\alpha \tag{4.2}$$

donde $\|\alpha\|_0$ es una pseudo-norma[1] l_0 que cuenta el número de elementos distintos de cero de α. Sin embargo, éste es un problema de optimización *NP-hard* y su resolución es muy compleja [28].

Afortunadamente, como apuntan Chen et al. [29], la norma L_1 también es capaz de resolver el problema de minimización de una manera *sparse* debido a la forma de la bola de la distancia

[1] A la norma L_0 se le denomina pseudo-norma porque no cumple la condición de norma $\| \lambda x \| = |\lambda| \, \| x \|$. En cambio para L_0, $\| \lambda x \|_0 = \| x \|_0$ para toda $\lambda \neq 0$.

inducida por la norma L_1. En la Figura 4.7 puede verse un ejemplo bi-dimensional de la comparación de las normas L_1 y L_2 y el porqué la norma L_1 conduce a una solución *sparse* mientras que la norma L_2 no.

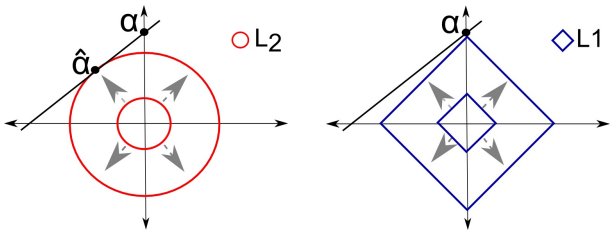

Figura 4.7: Norma L_1 (derecha) y norma L_2 (izquierda). La linea negra representa el conjunto de todas las soluciones. Usando minimización L_2, donde la frontera de la bola crece de una manera equidistante, se puede ver que la bola mínima que se encuentra con el set de posibles soluciones no arroja un resultado *sparse* ya que el punto aproximado yace en el espacio bi-dimensional no cero. La bola mínima L_1 encuentra el punto *sparse* que yace tan solo en el eje vertical.

Una vez decidido el uso de la norma L_1 para la aproximación del vector de coeficientes α, el problema queda definido entonces como:

$$\min_x \|\alpha\|_1 \text{ sujeto a } y = \Theta\alpha \tag{4.3}$$

con $\|\alpha\|_1 = |\alpha_1| + |\alpha_2| + \cdots + |\alpha_k|$ la suma de todos los coeficientes de α, por lo que una minimización conduce a una resolución *sparse*.

No obstante, la resolución del problema mediante la norma L_1 es equivalente a la resolución mediante la norma L_0 solo cuando la señal original es lo suficientemente *sparse* y se cumplen las condiciones para la matriz de medida mencionadas en el Capítulo 3. Debido a que para conjuntos de datos muy grandes la minimización de la norma L_1 puede resultar muy costosa computacionalmente, diversos trabajos se han centrado en desarrollar otros algoritmos de reconstrucción que pueden ser divididos en: optimización convexa [[30], [31], [32]], algoritmos voraces [[33], [34], [35]], y algoritmos combinatorios [[36], [37]]. Para un análisis más extenso de esta clasificación, puede consultarse el trabajo de Kutyniok [38].

En el desarrollo de este proyecto, de todos estos algoritmos se han escogido dos para realizar las reconstrucciones y comparar sus resultados: Un algoritmo voraz, Orthogonal Matching Pursuit (OMP) [50] y un algoritmo de optimización convexa, Least Absolute Shrinkage and Selection Operator (LASSO) [49]. Una explicación más extensa de estos dos algoritmos puede encontrarse en el Anexo C.

4.4.1. Detalles de la implementación

En la introducción de este capítulo se ha expuesto que, en el sistema de reconstrucción, en todo momento se trabaja con bloques de vídeo y parches de imagen fruto de la división de los originales en fragmentos más pequeños.

Para efectuar esta división, la opción más sencilla es simplemente dividirlos de una manera tradicional, por ejemplo, para el caso de imagen, separarla en parches de imagen del mismo tamaño. Sin embargo, en la implementación de este proyecto, siguiendo el trabajo de Hitomi [8],

esta división se realiza de manera solapada, tal y como se muestra en la Figura 4.8, originando $(M-p+1)\times(N-q+1)$ parches de imagen. De esta manera, cada píxel está presente en $p\times q$ parches diferentes de imagen y, por tanto, será reconstruido $p\times q$ veces, permitiendo después el cálculo del promedio entre estas reconstrucciones para obtener una mejor estimación para cada píxel.

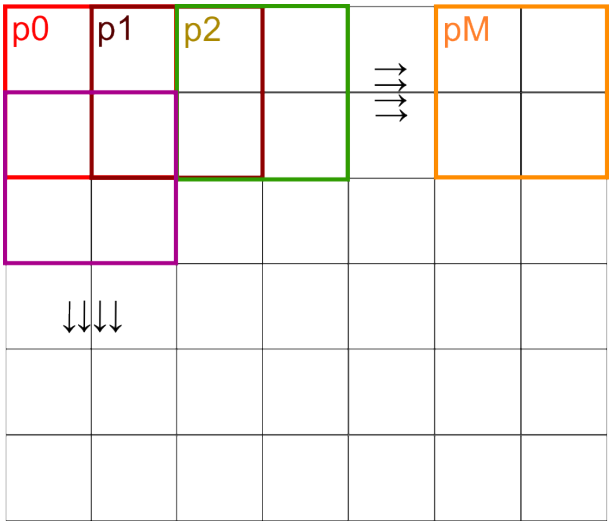

Figura 4.8: Subdivisión de la imagen Y de tamaño $M\times N$ en parches solapados, desplazando una ventana de tamaño, en este caso, 2×2, para su reconstrucción. La división genera $(M-p+1)\times(N-q+1)$ parches de imagen.

Notar que para realizar la composición del vídeo completo a partir de los bloques de vídeo según este método, para cada píxel, es necesario contar con todas las reconstrucciones en las que el píxel aparece. Por ello, los bordes de la imagen se pierden. Concretamente, para un bloque de vídeo de tamaño $M\times N\times T$, y un tamaño de bloque de $p\times q\times T$, el vídeo completo resultante será de tamaño $(M-2(p-1))\times(N-2(q-1))\times T$.

5. Adquisición de una base de datos

Para el desarrollo de este proyecto, se ha creado una base de datos de vídeos de alta velocidad y alta resolución temporal. Esta base de datos cuenta con vídeos de muy diferentes características, entre ellas: diferentes dispositivos de captura, para no obtener resultados adaptados a un único dispositivo de captura concreto; objetos de diferente naturaleza representados en las escenas, para obtener la mayor diversidad posible; y, lo más importante, diferentes características espaciales y temporales, esto es, movimientos bruscos, suaves, periódicos, vídeos con muchos contornos, etc.

Para una versión en baja resolución y comprimida de esta base de datos consultar el Anexo D. Es importante recalcar que los experimentos no pueden reproducirse con las versiones en baja resolución de los vídeos. La base de datos de alta resolución esta disponible bajo petición, pero, debido al tamaño de los vídeos sin comprimir y la limitación en recursos de almacenamiento on-line, facilitarla completa ha sido inviable.

Los vídeos de esta base de datos provienen de tres fuentes distintas, que se describen a continuación.

Parte de estos vídeos, han sido capturados con el equipamiento del instituto universitario de investigación en ingeniería de Aragón (I3A). La captura se realizó con el montaje mostrado en la Figura 5.1 con una cámara Photron SA2 [39]. Durante este proceso, comprobamos de primera mano algunas de las limitaciones de los equipos de captura de vídeo de alta velocidad. Para el proceso de captura se necesita una fuente de luz potente, así como estabilizar tanto la cámara como el objeto a grabar. Además, la profundidad de campo de la cámara era muy limitada, por lo que el objeto a grabar tiene que estar situado en un punto muy concreto, de otra manera, aparece desenfocado. Esto pone de manifiesto, una vez más, la necesidad de una alternativa a la captura convencional, ya que a pesar de tratarse de equipamiento especializado, todavía presenta ciertas limitaciones. Durante el proceso de captura, se grabaron **12** vídeos con una resolución temporal de 1000 fps (fotogramas por segundo) y con una resolución espacial de 1024×1024 píxeles.

Otros **5** vídeos han sido proporcionados por el departamento de ingeniería informática de la universidad Rey Juan Carlos (URJC), y están capturados a 1000 fps y con una resolución de 1280×1024 píxeles.

Por último, los restantes **9** vídeos son parte de los vídeos usados en el trabajo de Hitomi et al. [8]. Están capturados a 30 fps y con una resolución de 352×288. Es importante destacar que estos vídeos son de resoluciones frecuenciales y espaciales muy diferentes al resto de los vídeos de la base de datos. El uso de estos vídeos se debe a que queríamos comprobar qué sucede al

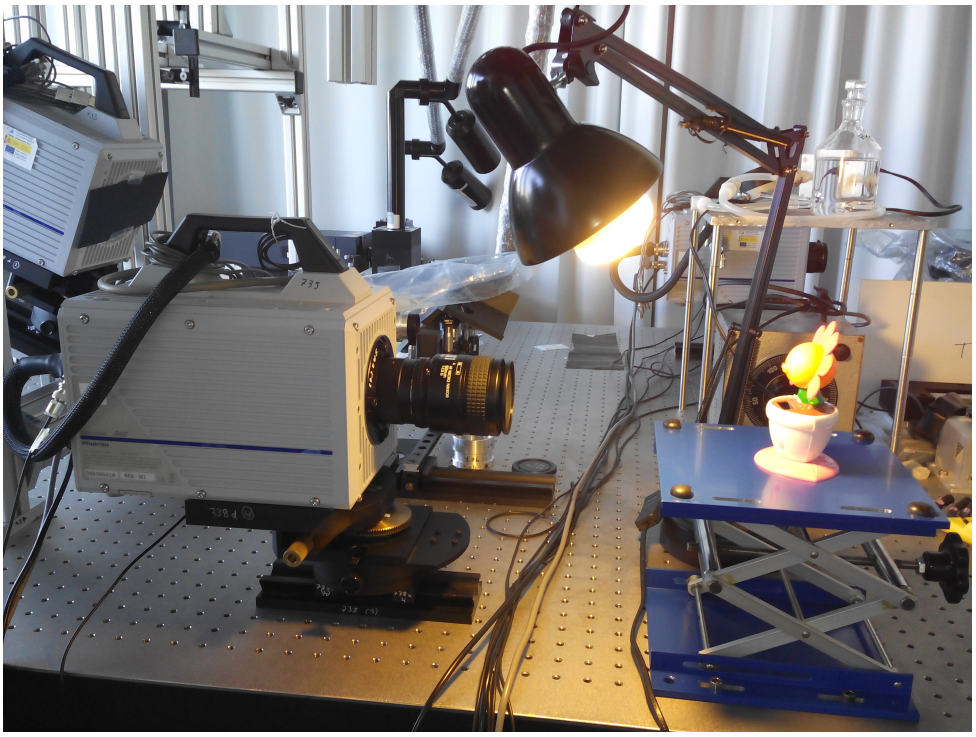

Figura 5.1: Montaje para la captura de vídeo con la cámara Photron SA2 en el I3A consistente en un soporte tanto para la cámara como para el objeto a grabar, y una fuente de luz iluminando el objeto.

intentar reconstruir vídeos capturados con características muy diferentes de los vídeos con los que se entrena el diccionario, es decir, comprobar cuanto de extrapolable es dicho diccionario.

De estos 26 vídeos, diez se han usado para el entrenamiento de diccionarios. De estos diez, ocho pertenecen a los vídeos capturados en el I3A y los otros dos a los vídeos proporcionados por la URJC. Una muestra de los vídeos de entrenamiento puede verse en el Anexo D.

Se ha realizado un escalado de los vídeos tanto para entrenamiento como para reconstrucción, debido a que el tiempo de cálculo se incrementa con la dimensionalidad y el tamaño de los datos, y el procesado de vídeos de mayor tamaño durante este proyecto se hizo inviable debido a la limitación de recursos. Para la parte de entrenamiento, los vídeos se han escalado a una resolución espacial de 512×512 y se han extraído 200 fotogramas de cada vídeo, esto es, cada vídeo de entrenamiento es de tamaño $512 \times 512 \times 200$. Para la parte de reconstrucción, los vídeos se han escalado a una resolución espacial de 200×200 y el número de fotogramas extraídos depende del tipo de análisis realizado, así pues se tienen vídeos de reconstrucción de 200×200 y 10, 20 y 30 fotogramas, ya que éstas son las longitudes de vídeo para las que se analiza la reconstrucción.

6. Análisis de parámetros

6.1. Introducción

En un sistema basado en *compressive sensing*, independientemente de la aplicación final, la calidad de los resultados obtenidos depende en gran medida de un elevado número de factores, entre los que se incluyen una serie de parámetros relativos al algoritmo de entrenamiento del diccionario a partir de la base de datos, el tipo de algoritmo de reconstrucción, o la matriz de medida utilizada. Con el objetivo de dar con los valores óptimos de estos parámetros a emplear en nuestro sistema, se han realizado una serie de pruebas y análisis, que se exponen, junto con las conclusiones obtenidas en cada caso, en este capítulo.

Asimismo, observamos que la calidad de la reconstrucción de un vídeo exhibe una dependencia significativa con las características del vídeo a reconstruir, y en particular con sus frecuencias espaciales y temporales. Es por esto que los seis vídeos utilizados en este análisis de parámetros se han estudiado, analizando su variación espacio-temporal, en la Subsección 6.2.

Métricas de error. Para cada uno de los parámetros de influencia que se estudiarán, se han analizado los seis vídeos mencionados, calculando el error obtenido en la reconstrucción de cada uno mediante las métricas de error PSNR (dB) [40] y MS-SSIM [41]. PSNR (Peak Signal-to-Noise Ratio) es una métrica objetiva comúnmente utilizada en procesado de señal, basada en la diferencia entre píxeles correspondientes de ambas secuencias de vídeo. La expresión matemática para el cálculo de la misma, que da el resultado en decibelios, se muestra en las siguientes ecuaciones:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \| I(i,j) - K(i,j) \|^2 \tag{6.1}$$

$$PSNR = 10 \log_{10}(\frac{MAX_I^2}{MSE}) = 20 \log_{10}(\frac{MAX_I}{\sqrt{MSE}}) \tag{6.2}$$

MS-SSIM (Multi-Scale Structural Similarity Index Metric) es una métrica muy utilizada en visión por computador para la medida de similitud entre dos imágenes. Esta métrica tiene en cuenta el modo en que funciona el sistema visual humano a la hora de calcular el error, y se diseñó para mejorar las métricas tradicionales de error como MSE o PSNR basadas sólo en la diferencia entre píxeles, y por tanto inconsistentes en muchas ocasiones con la percepción del sistema visual humano. La métrica MS-SSIM se calcula sobre varias ventanas de la imagen y

para diferentes resoluciones. La medida de dos ventanas x e y del mismo tamaño $N \times N$ se obtiene a partir de la comparación de la luminancia l, del contraste c, y de la estructura s. La luminancia se compara a una sola escala (la original, denotada M), pero el contraste y la estructura se comparan en múltiples (M) escalas. Para ello, se aplica iterativamente un filtro paso bajo y un diezmado de la imagen filtrada por un factor 2. La métrica se calcula por tanto con la siguiente fórmula:

$$MSSSIM(\mathbf{x},\mathbf{y}) = [l_M(\mathbf{x},\mathbf{y})]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(\mathbf{x},\mathbf{y})]^{\beta_j} [s_j(\mathbf{x},\mathbf{y})]^{\gamma_j} \tag{6.3}$$

donde las diferencias de luminancia (l), de contraste (c) y en la estructura (s) se calculan como sigue:

$$\begin{aligned} l(\mathbf{x},\mathbf{y}) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2\mu_y^2 + C_1} \\ c(\mathbf{x},\mathbf{y}) &= \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2\sigma_y^2 + C_2} \\ s(\mathbf{x},\mathbf{y})) &= \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \end{aligned} \tag{6.4}$$

siendo μ_x, μ_y las respectivas medias de las ventanas x,y; σ_x, σ_y las desviaciones típicas y $\sigma_{x,y}$ su covarianza. Los valores C_1, $C_2 2$ y C_3 son constantes para estabilizar la división donde $C_1 = (K_1 L)^2$, $C_2 = (K_2 L)^2$ y $C_3 = C_2/2$, siendo L el rango dinámico de los valores de píxel ($L = 2^{bitsperpixel} - 1$). Los exponentes α_M, β_j y γ_j se usan para ajustar la importancia relativa de diferentes componentes. Las fórmulas de la Ecuación 6.4 hacen que la diferencia se compute en valores relativos en vez de absolutos, emulando una de las características del funcionamiento del sistema visual humano.

Parámetros por defecto. En todas las pruebas que se exponen en las subsecciones siguientes de este capítulo se ha variado uno de los parámetros del sistema, dejando todos los demás fijos, y analizando la influencia de dicho parámetro. Para poder hacer esto, se han fijado un conjunto de parámetros que denominamos "por defecto", que son los siguientes:

- Bloques de vídeo de tamaño $7 \times 7 \times 20$ píxeles.
- Diccionario de 10000 elementos entrenado con el algoritmo K-SVD.
- Algoritmo de reconstrucción Lasso.
- Muestreo con obturador *pixel-wise*

Vídeos para el análisis. Se han utilizado para el análisis seis vídeos de escenas de muy distinta naturaleza, con diferentes objetos, movimientos variados, y por tanto de diferentes características espacio-temporales. Éstas se analizan en la Sección 6.2, dada su influencia en la calidad de la reconstrucción final. Un fotograma representativo de cada uno de los vídeos se muestra en la Figura 6.1. Naturalmente, ninguno de los vídeos que se han utilizado en este análisis forma parte de la base de datos empleada para el entrenamiento de diccionarios.

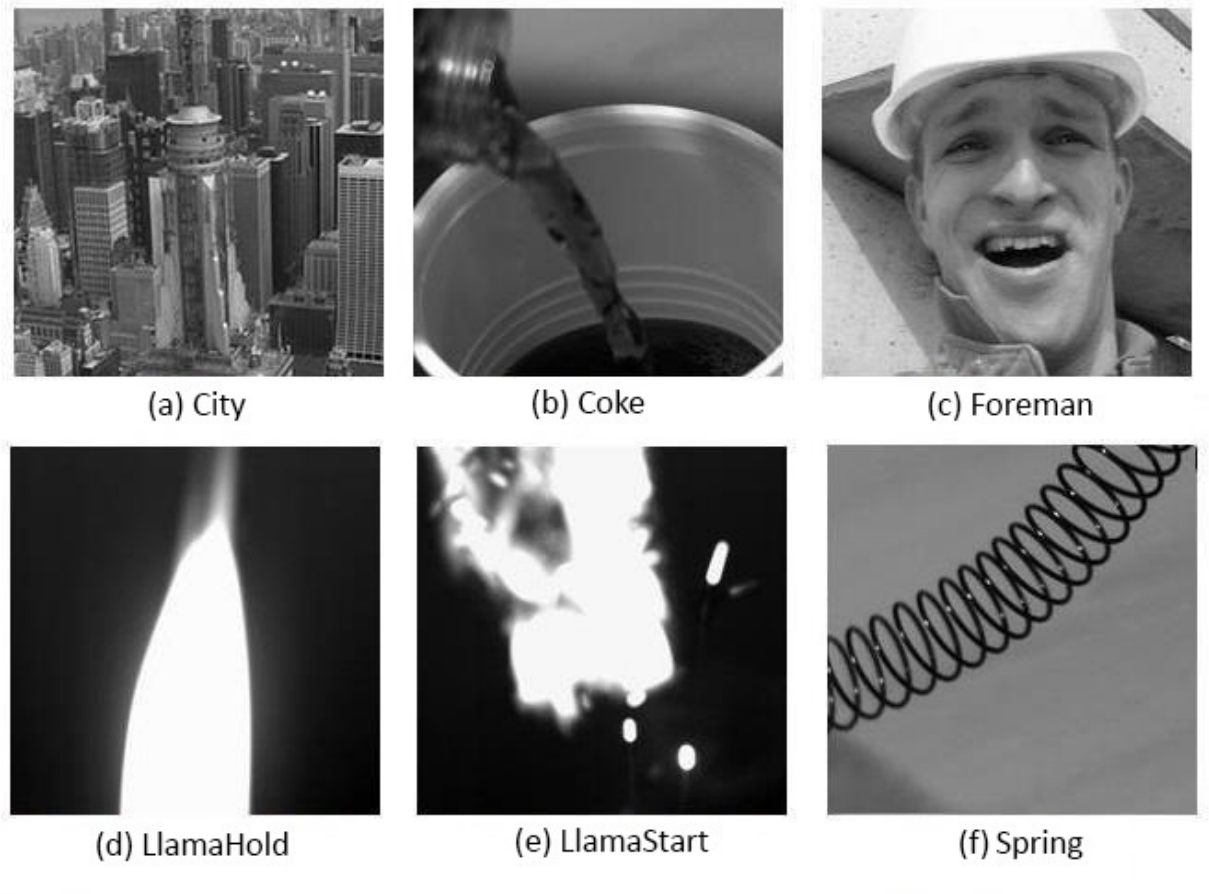

Figura 6.1: Fotograma representativo extraído de cada uno de los vídeos utilizados en el análisis.

6.2. Características espacio-temporales de los vídeos usados en el análisis

Es interesante hacer una caracterización de estos vídeos porque el resultado de la reconstrucción, como se ha mencionado anteriormente, depende fuertemente de las características del vídeo que se desea reconstruir, como por ejemplo su distribución de frecuencias espaciales (si tiene muchos contornos o es más bien suave), o su distribución de frecuencias temporales (si ocurre mucho movimiento o poco). Para la caracterización de imagen, existen muchos procesos estandarizados, sin embargo, la caracterización de un vídeo de la manera que nos interesa en este proyecto, no es un proceso tan sencillo. Para intentar dar con un método de caracterización que nos permita saber con antelación si un vídeo se va a poder reconstruir correctamente, esto es, caracterizarlo, se han explorado diferentes métodos que se listan a continuación:

- *Histograma de ratios de densidad de energía de alta frecuencia temporal calculados para cada bloque, tras dividir el vídeo en bloques del tamaño del que se quiere reconstruir* ($\mathbf{H}_{ratios}$). Estos ratios se calculan realizando una DFT unidimensional a lo largo del eje temporal para cada píxel del bloque. Después, se suman los coeficientes de la DFT de todos los píxe-

les del bloque y se calcula el ratio entre la energía total y la energía de alta frecuencia de esta DFT conjunta. Este valor calculado para todos los bloques que componen el vídeo es lo que se representa en el histograma para después calcular sus descriptores estadísticos. El objetivo de esta caracterización es obtener una medida de la variación temporal del bloque de vídeo. Obtenido el histograma, se calculan sus descriptores estadísticos: media, desviación típica, skewness y kurtosis.

- *Histograma de las varianzas calculadas para cada bloque, tras dividir el vídeo en bloques del tamaño del que se quiere reconstruir* ($\mathbf{H}_{varianzas}$). Las varianzas se calculan sobre el bloque completo de vídeo y se representan en el histograma para después calcular sus descriptores estadísticos. El objetivo de esta caracterización es obtener una medida conjunta de la variación espacial y temporal para tener en cuenta tanto los eventos temporales (e.g. movimiento) como los espaciales (e.g. contornos). Una vez obtenido el histograma, se calculan los mismos descriptores estadísticos que en el caso anterior: media, desviación típica, skewness y kurtosis.

- *Suma de la diferencia entre fotogramas* ($\mathbf{S}_{diff}$). Para esta medida se calcula una imagen diferencia que contiene la suma de todas las diferencias entre fotogramas consecutivos. Después, se suman todos los píxeles de esta imagen obteniendo un único valor. El objetivo es obtener una medida de cuánto movimiento (cuánta variación espacio-temporal) hay en el vídeo completo.

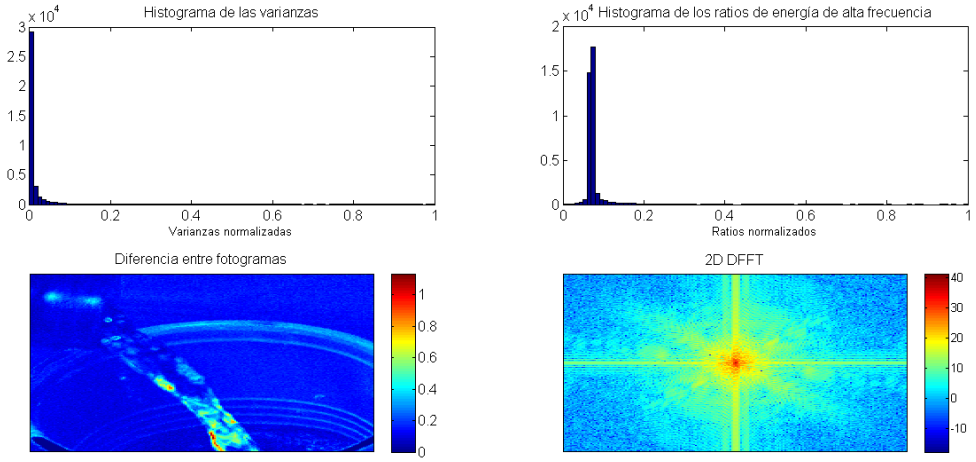

Figura 6.2: Métodos de caracterización para el vídeo Coke. En la fila de arriba se pueden ver el histograma de las varianzas de cada bloque (izquierda) y el histograma del ratio de energía de alta frecuencia temporal (derecha). En la fila de abajo se puede ver la imagen calculada como la suma de diferencias entre fotogramas consecutivos, donde se puede ver que este vídeo presenta muy poco movimiento, y la transformada de Fourier bi-dimensional.

Estos métodos de caracterización se han obtenido para los seis vídeos de test. La Figura 6.2 muestra, para el vídeo *Coke*, el resultado de los tres métodos de caracterización y la transformada discreta de Fourier bi-dimensional. Estos mismos resultados para los cinco vídeos de test restantes se pueden encontrar en el Anexo E. Una vez obtenidos los histogramas y la diferencia entre fotogramas, se calculan los descriptores de los histogramas mencionados y la suma de la diferencia de fotogramas. El objetivo es ver cuál de estos descriptores o valores es una buena forma de

Vídeo	μ	σ	sk	k
City	$0,3323$	$0,1273$	$0,9124$	$4,7346$
Coke	$0,0764$	$0,0381$	$11,8942$	$196,4976$
Foreman	$0,2611$	$0,0976$	$1,5505$	$8,059$
LlamaHold	$0,6749$	$0,0855$	$-2,62$	$15,9144$
LlamaStart	$0,2694$	$0,1943$	$1,1941$	$3,5215$
Spring	$0,2016$	$0,2129$	$1,4616$	$3,9331$

Tabla 6.1: Media μ, varianza σ, skewness sk y kurtosis k correspondientes a los histogramas mostrados en la Figura 6.3.

caracterizar la variación espacio-temporal de un vídeo dado para, dado un vídeo, poder inferir cómo de buena será la reconstrucción de una escena de ese tipo. Para ello, se reconstruyen los seis vídeos de test (usando los parámetros "por defecto"mencionados en la Sección 6.1), se mide su calidad utilizando PSNR y MS-SSIM, y se compara esta calidad de los resultados con las medidas dadas por los métodos de caracterización descritos en este apartado.

Para ello, se calculó la correlación entre métricas calidad y medidas de caracterización utilizando los coeficientes de correlación de Pearson [42] y Spearman [43]. La medida de caracterización que arroja una mayor correlación con la calidad final es la desviación típica del histograma de ratios de energía de alta frecuencia temporal, σ_{ratios}. Para esta medida, el coeficiente de Pearson arroja un valor $\rho_P = -0{,}9636$ con un p-valor de 0,0020, mientras que el coeficiente de Spearman resulta en un valor $\rho_S = -1$ y un p-valor de 0,0028. Por tanto, σ_{ratios} es el descriptor que se ha decidido utilizar para la caracterización de los vídeos. En la Figura 6.3 puede verse el histograma de los ratios de energía de alta frecuencia temporal $\mathbf{H}_{ratios}$ para cada uno de los seis vídeos de test, mientras que en la Tabla 6.1 se presenta la media, desviación típica, skewness y kurtosis para cada uno de estos seis histogramas.

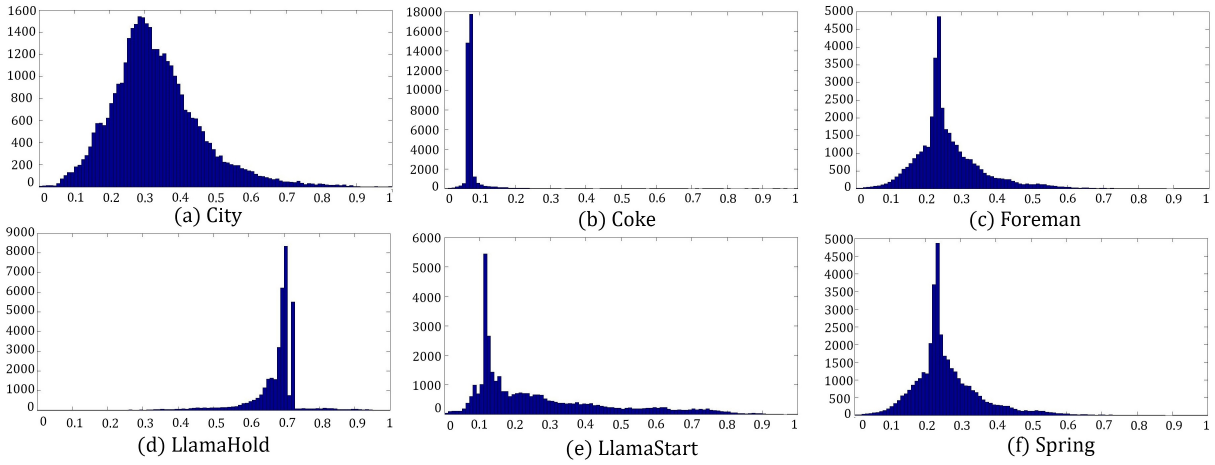

Figura 6.3: Histograma de los ratios de energía de alta frecuencia temporal $\mathbf{H}_{ratios}$ para cada uno de los vídeos utilizados en el análisis.

Los valores de PSNR [40] y MS-SSIM [41] obtenidos para cada uno de los seis vídeos pueden verse en la Figura 6.4, mientras que en la Figura 6.5 se muestra un fotograma del resultado reconstruido para cada vídeo. Como se puede ver, el resultado no siempre es bueno, y depende

en gran medida del vídeo que se quiere reconstruir: vídeos con histogramas muy esparcidos frecuencialmente—esto es, un elevado valor de σ_{ratios}—tendrán peores reconstrucciones, siendo por tanto σ_ratios el mejor indicador de la calidad de los resultados que se obtendrán según las características del vídeo a reconstruir.

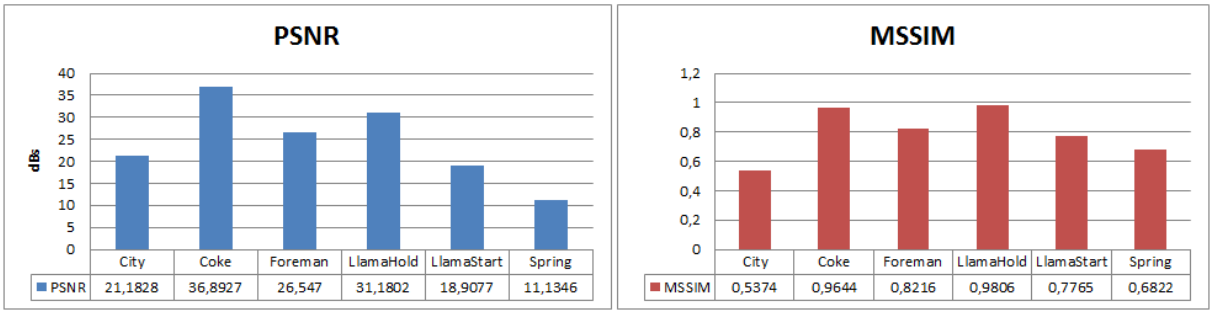

Figura 6.4: PSNR (izquierda) y MS-SSIM (derecha) para cada uno de los vídeos reconstruidos con el diccionario por defecto.

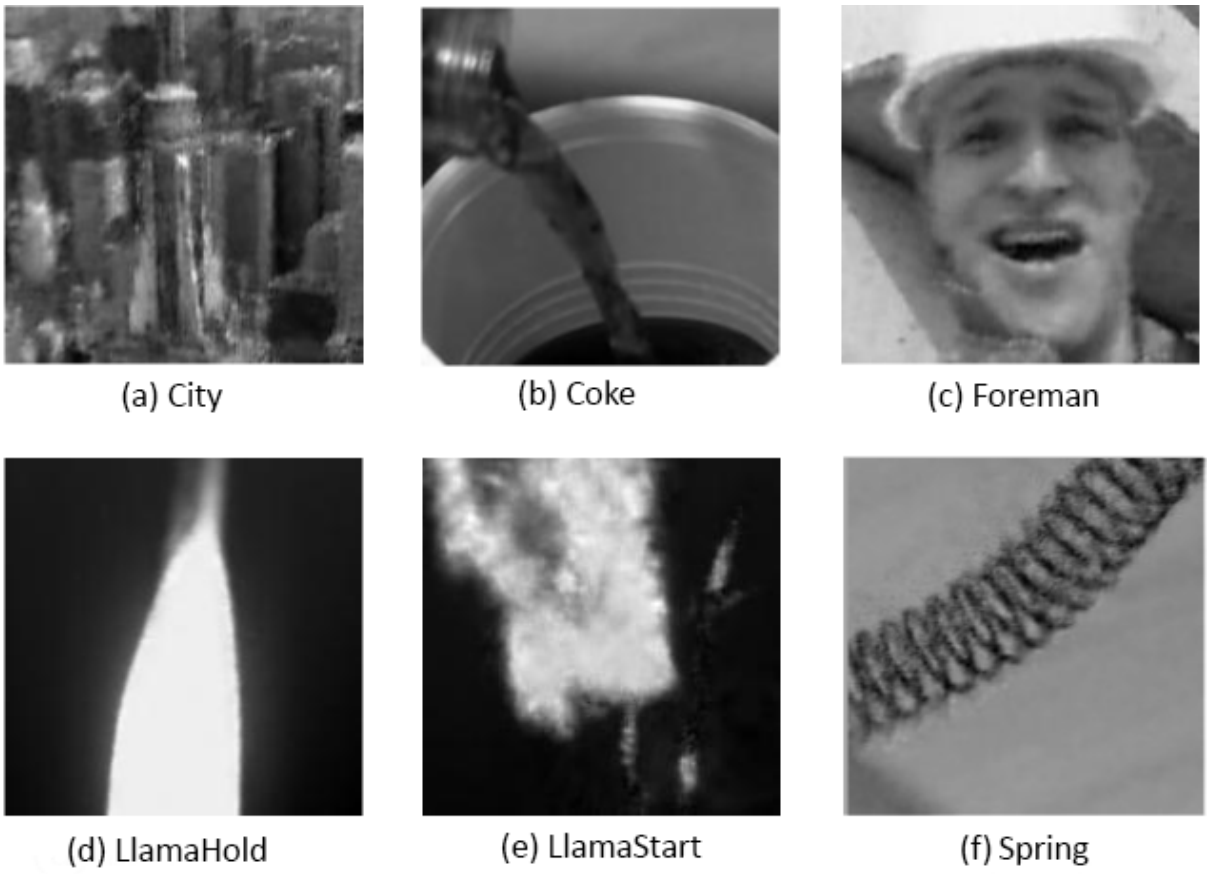

Figura 6.5: Fotograma reconstruido para los seis vídeos utilizados en el análisis con el diccionario por defecto. Se puede ver como la calidad de los resultados depende mucho del vídeo que se quiera reconstruir y el resultado visual coincide con los resultados de PSNR y MS-SSIM obtenidos.

6.3. Selección del tamaño de bloque de vídeo

Como se ha explicado en el Capítulo 4, es necesario dividir un vídeo en bloques para su reconstrucción. Así pues, se reconstruirá el vídeo bloque a bloque, en bloques de dimensiones $p \times q \times T$, en este caso, $7 \times 7 \times 20$, que luego se unirán para formar el vídeo reconstruido final. Estos bloques pueden verse como pequeños vídeos en sí mismos, de corta duración y pequeño tamaño.

La influencia de las dimensiones espaciales y temporales de estos bloques en la calidad de los resultados reconstruidos se analiza a continuación. El tamaño del bloque influye también en el tiempo de cálculo de la reconstrucción, pudiendo generar un compromiso entre calidad y tiempo de ejecución.

6.3.1. Tamaño espacial del bloque de vídeo

Elegimos bloques cuadrados, para reducir el número de posibles combinaciones, de tamaños $p = q = \{5, 7, 9\}$. Tamaños más elevados son computacionalmente inviables y contienen demasiados detalles, por lo que el algoritmo de entrenamiento se sobre-adapta a ellos, mientras que tamaños menores (e.g. 3×3 píxeles) apenas contienen información en sí mismos y no capturan la estructura subyacente.

Las Figuras 6.6 y 6.7 muestran el error de la reconstrucción en los seis vídeos de test para diferentes valores de p y q, medido con MS-SSIM y PSNR respectivamente.

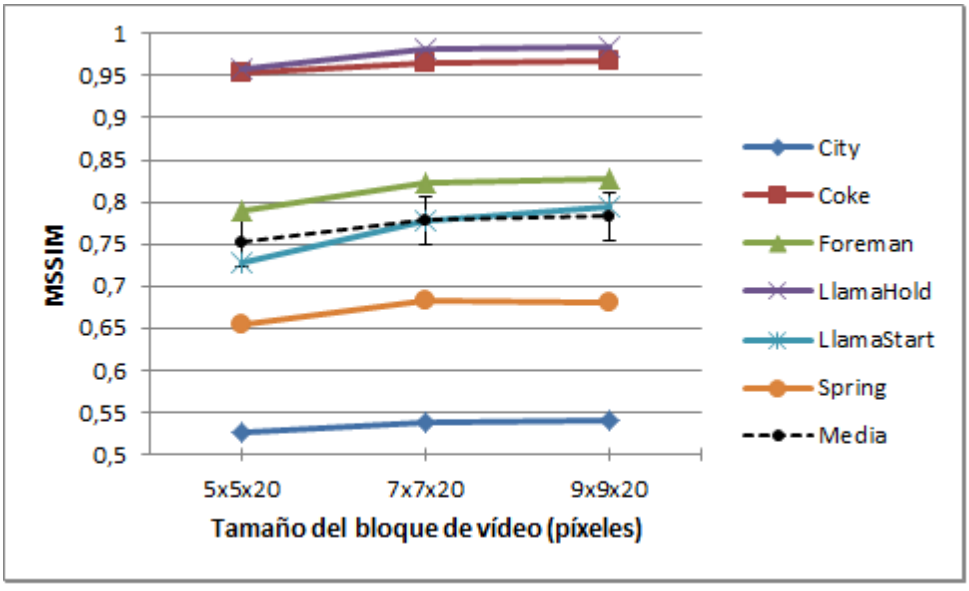

Figura 6.6: MS-SSIM de la reconstrucción en función del tamaño espacial del bloque

En primer lugar, observamos una variabilidad del error de reconstrucción con el tipo de vídeo, tal y como se ha explicado en la sección anterior. En cuanto al tamaño de bloque, se puede ver un incremento tanto del PSNR como del MS-SSIM para todos los vídeos en el paso de bloques de 5×5 píxeles a 7×7 píxeles mientras que el aumento a 9×9 píxeles no tiene una

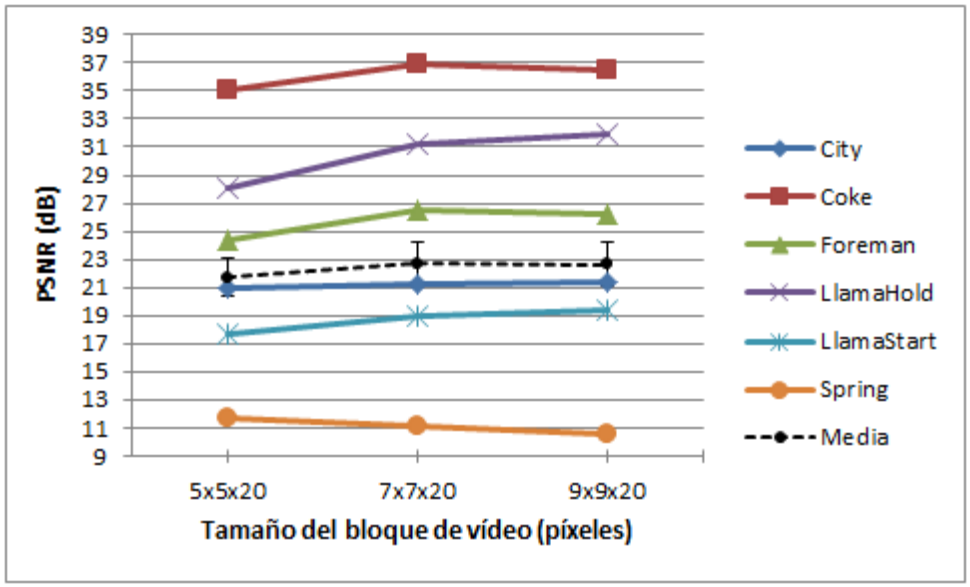

Figura 6.7: PSNR de la reconstrucción en función del tamaño espacial del bloque

mejoría clara. Esto es así porque al aumentar mucho el tamaño del parche espacial se incluyen muchos detalles y características muy concretas de la imagen que probablemente se perderán en la reconstrucción.

Además, en la Figura 6.8 se observa el incremento de tiempo de cálculo de la reconstrucción al incrementar el tamaño de bloque, que es muy elevado. Así pues, considerando la calidad del resultado y tiempo de cálculo se concluye que el mejor tamaño de bloque espacial es 7×7 píxeles.

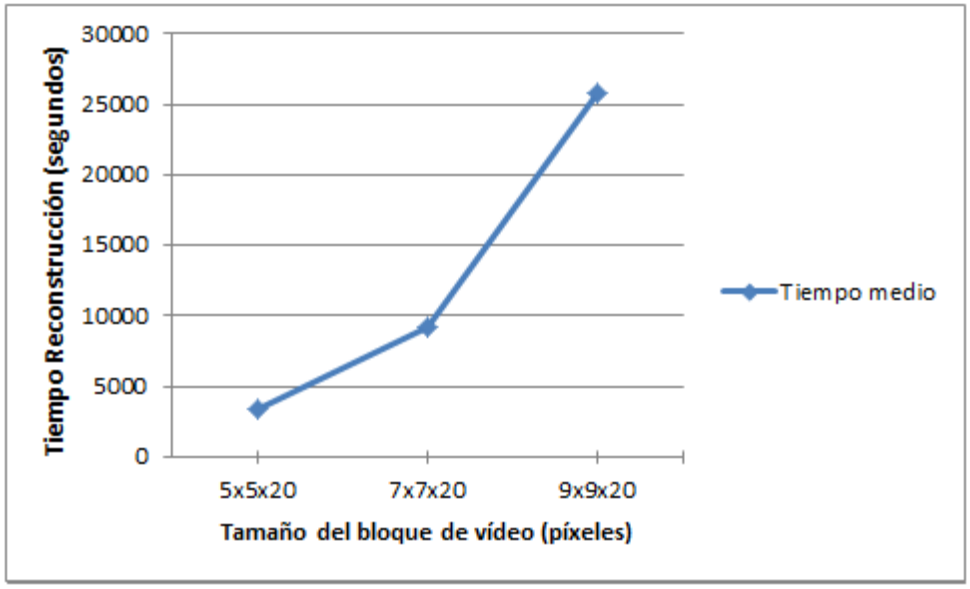

Figura 6.8: Tiempo medio de la reconstrucción del vídeo en función del tamaño espacial del bloque. Nótese que este tiempo se corresponde con el tiempo de la reconstrucción del vídeo completo y no de un solo bloque.

6.3.2. Tamaño temporal del bloque de vídeo

Este parámetro, T, determina el número de fotogramas que contiene cada bloque de vídeo. Es por tanto el número de fotogramas que luego se van a reconstruir a partir de una única imagen codificada. Así, se puede ver ya de forma intuitiva que a mayor número de fotogramas (mayor T), peor va a ser la calidad de la reconstrucción ya que en en una misma imagen se deberá codificar información para reconstruir más fotogramas. Para este parámetro hemos tomado tres valores, $T = \{10, 20, 30\}$, ya que valores mayores ofrecen una calidad de reconstrucción muy baja, y valores menores requieren una fragmentación excesiva del vídeo final en un elevado número de bloques.

En las Figuras 6.9 y 6.10 se muestra el error de los vídeos reconstruidos respecto a los originales para los tres valores de T medido con MS-SSIM y PSNR, respectivamente. De acuerdo con lo esperado, la calidad de los resultados disminuye a mayor número valor de T de forma prácticamente lineal (aunque con diferentes pendientes para cada vídeo).

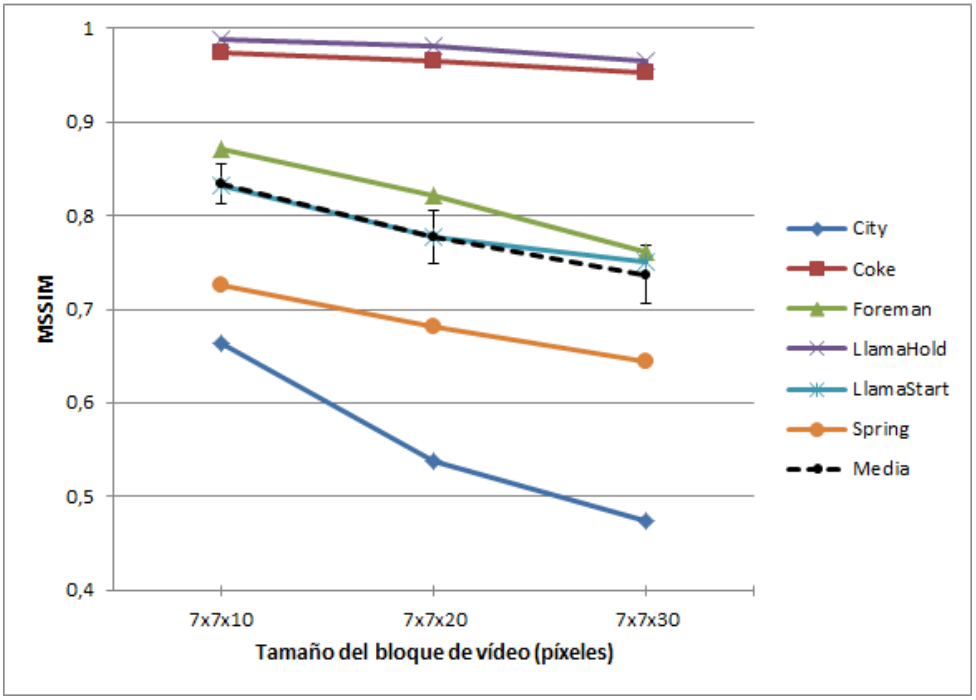

Figura 6.9: MS-SSIM de la reconstrucción en función del tamaño temporal del bloque.

En términos del tiempo de ejecución, la Figura 6.11 muestra el incremento de éste al aumentar el tamaño del bloque en la dimensión temporal.

Así, mayores tamaños de bloque en la dimensión temporal tienden a ofrecer peores resultados en términos de error y tiempo de reconstrucción. Sin embargo, al hacer la elección se debe tener en cuenta también que un número bajo supone una gran fragmentación del vídeo. En este caso, se ha optado por la decision de 20 fotogramas, ya que el decrecimiento del PSNR y MS-SSIM todavía se encuentra en unos valores aceptables y el incremento del tiempo no es significativo.

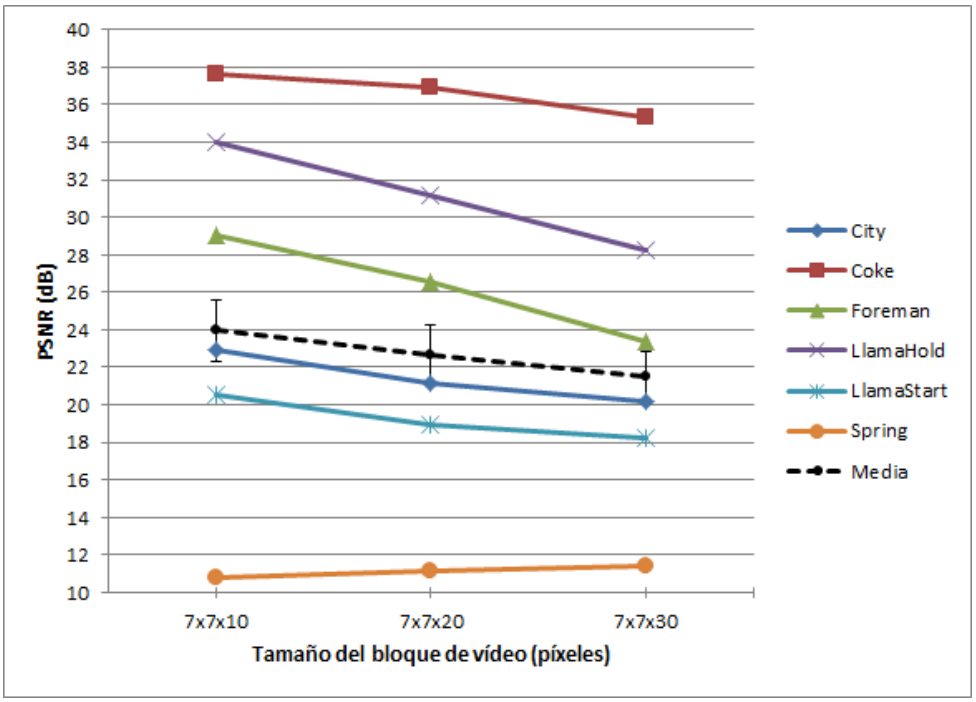

Figura 6.10: PSNR de la reconstrucción en función del tamaño temporal del bloque.

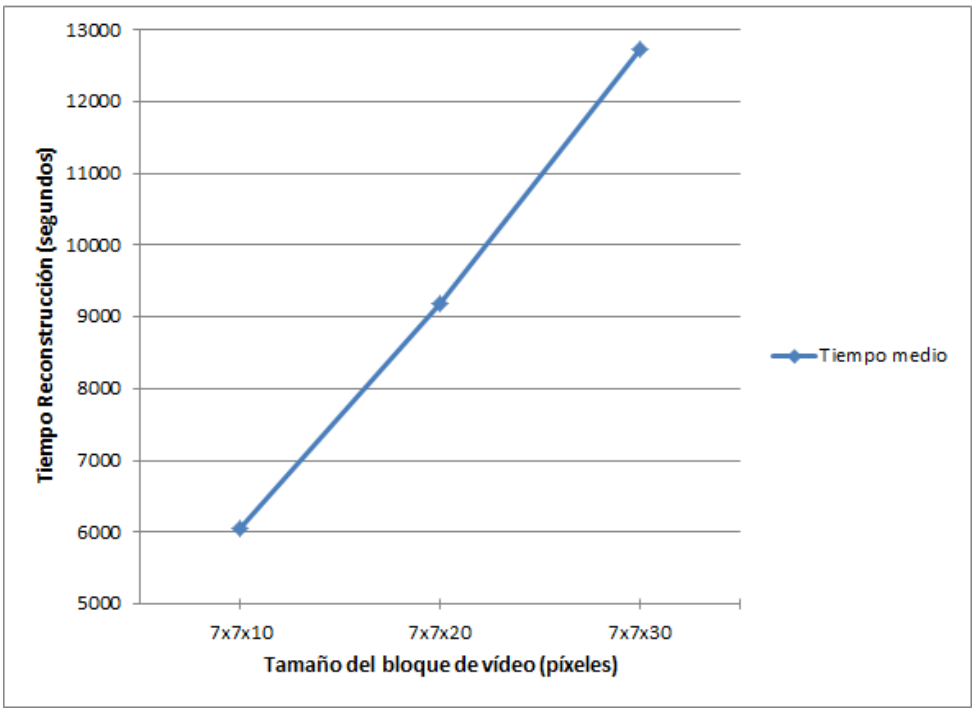

Figura 6.11: Tiempo medio de la reconstrucción del vídeo en función del tamaño temporal del bloque. Nótese que este tiempo se corresponde con el tiempo de la reconstrucción del vídeo completo y no de un solo bloque.

6.4. Diccionario

La elección o creación de un diccionario adecuado al tipo de señal (3D en este caso, al ser vídeos) que queremos reconstruir es crucial para la obtención de reconstrucciones de calidad

utilizando *compressive sensing*. En esta sección, primero evaluamos la necesidad de utilizar un diccionario aprendido, frente al uso de diccionarios (o bases) comúnmente utilizadas en *compressive sensing* como DCT (Transformada Discreta del Coseno o Discrete Cosine Transform). Después proponemos y analizamos distintas estrategias de selección de bloques de vídeo de la base de datos a partir de los cuales se va a entrenar el diccionario.

6.4.1. Diccionario entrenado vs DCT

Es una práctica habitual, en especial cuando se trata con información visual dentro de un marco de *compressive sensing*, utilizar la transformada discreta del coseno (DCT de aquí en adelante) como diccionario para la reconstrucción de información. De hecho, es bien conocido que la compresión jpeg [7] utiliza DCT para representar las señales de forma comprimida (de manera análoga a lo que se hace en *compressive sensing* al tratar de representar la señal con un número de muestras reducido). Es por esto que evaluamos aquí la adecuación de DCT para representar las señales de vídeo de alta velocidad que tratamos de reconstruir en este trabajo. Además, la comparamos con el uso de un diccionario entrenado con nuestros parámetros "por defecto", que está entrenado específicamente a partir de vídeos de alta resolución temporal.

Al tratarse de señales de vídeo, debe construirse una base DCT en tres dimensiones. Para poder hacer equiparables los dos diccionarios y que la comparación sea justa, se ha comparado el diccionario DCT con un diccionario entrenado de 980 elementos (en lugar de los 10000 del diccionario por defecto), porque 980 es el número de elementos de un diccionario DCT para un tamaño de bloque de $7 \times 7 \times 20$.

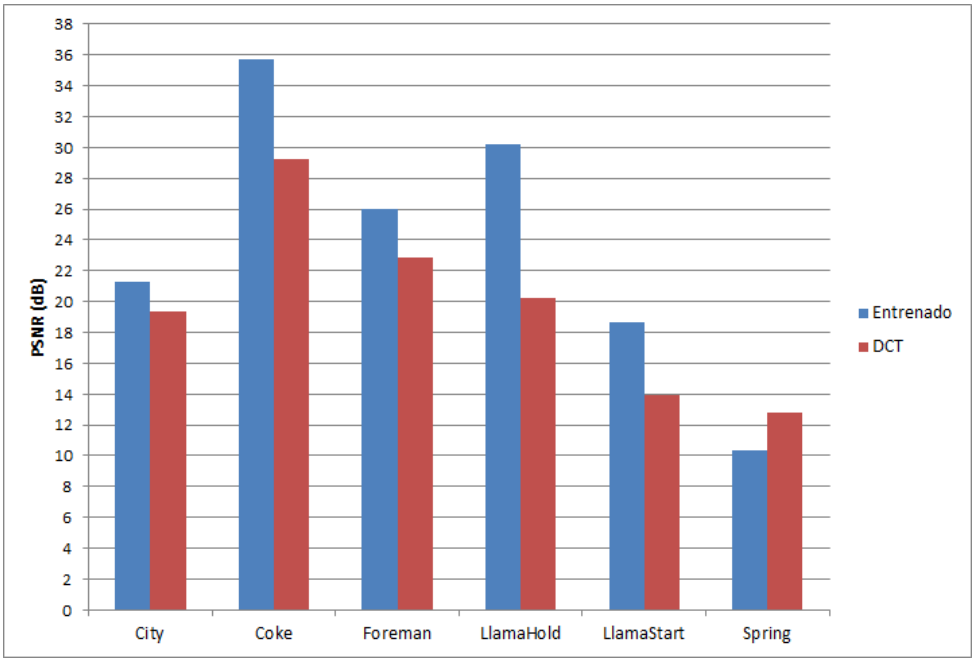

Figura 6.12: Comparación del PSNR de los resultados utilizando un diccionario aprendido o el diccionario DCT.

La Figura 6.12 muestra la comparativa entre ambos diccionarios para los seis vídeos de la evaluación. En general se obtienen mejores resultados con un diccionario adaptado al conjunto

de datos de interés. Se produce una excepción con el vídeo *Spring*, que probablemente se deba a ruido o simplemente sea un resultado arbitrario, ya que la reconstrucción lograda es de baja calidad y una diferencia de 2 dB no puede asegurarse que sea causada por la diferencia de diccionario. Es importante notar que los vídeos de la base de datos de entrenamiento son de naturaleza similar, pero representan escenas variadas y tienen diferente resolución espacial y temporal respecto a los vídeos de test. Si tomamos, por ejemplo, *Foreman* o *City*, ambos son vídeos de menor resolución temporal que los de la base de datos y que además representan escenas (una cara y una ciudad) que no se asemejan nada a las de la base de datos; es en estos casos en los que la diferencia entre usar DCT o usar un diccionario es menor. Así pues, el algoritmo de entrenamiento logra generar un diccionario extrapolable a otros vídeos, pero esta capacidad de extrapolación es limitada, indicando que la naturaleza de la base de datos ha de ser similar a aquella de los vídeos que queremos reconstruir para obtener resultados de alta calidad.

6.4.2. Método de selección de bloques de entrenamiento

Para el entrenamiento del diccionario, se usan un número de vídeos divididos en bloques de tamaño correspondiente al tamaño de átomo (i.e., bloque de vídeo) deseado. Con este método se obtiene un número de bloques de entrenamiento muy elevado con el que, debido al tiempo de cálculo, es inviable entrenar el diccionario. El número de bloques con el que se quiere entrenar el diccionario son 70000, que corresponde aproximadamente a un 13 % de los bloques totales generados a partir de los vídeos de entrenamiento, por lo que se debe seleccionar un conjunto de bloques del conjunto total siguiendo algún criterio. La solución más directa sería escogerlos aleatoriamente; sin embargo, esta solución podría no ser óptima, ya que de todo este conjunto de bloques, hay una cantidad muy elevada de bloques sin información (en los que no ocurre nada durante la secuencia) y unos pocos con gran cantidad de información (donde hay más detalles o se produce más movimiento).

Con el objetivo de mejorar los resultados, se han explorado diferentes tipos de muestreo del conjunto original para intentar obtener una distribución óptima de los bloques de entrenamiento. Esto supone, por un lado, evitar que una gran parte de los bloques sean uniformes y/o muy similares entre sí y no se capturen los detalles y características, y por otro lado, que la distribución de bloques seleccionada se aproxime a la distribución original de bloques (i.e. si en la distribución original había un gran número de bloques de frecuencias espaciales medias, respetar esto en la selección realizada para el entrenamiento). Los métodos que proponemos y evaluamos son:

- **Muestreo aleatorio**: Se seleccionan el número de muestras aleatorias deseadas del conjunto original.
- **Muestreo por selección de varianza**: Consiste en calcular la varianza de los bloques del conjunto original, agruparlos en categorías (baja, media y alta varianza) y posteriormente seleccionar aleatoriamente el mismo número de bloques de cada categoría. Con esto se pretende tener un conjunto variado de bloques de entrenamiento y asegurar que se seleccionan bloques de alta varianza, es decir, con grandes cambios, ya sean movimiento o contornos.
- **Muestreo estratificado basado en la curva gamma**: Por curva gamma se entiende la curva comúnmente utilizada en el campo de la imagen: la curva de corrección gamma,

que se define como $f(x) = x^\gamma$, que para valores de $\gamma < 1$ es monótonamente creciente y geométricamente cóncava. El objetivo al usar esta curva gamma es, tras ordenar los bloques en orden creciente de varianza normalizada[1], lograr seleccionar más cantidad de bloques de la zona de alta varianza. A este efecto, se analiza el efecto de dos casos: $\gamma = 0,7$ que proporciona una curva más próxima a un muestreo *lineal* y $\gamma = 0,3$ que proporciona una curva más cóncava (geométricamente). Se utiliza, mediante esta curva, muestreo estratificado. El objetivo del muestreo estratificado es asegurar que habrá representantes de todos los estratos y tener más variabilidad. Para ello, se divide el rango uniformemente en el número de muestras deseado, y aplicando la función de la curva gamma a estas divisiones se calculan umbrales de acuerdo a la curva gamma. Hecho esto, se selecciona aleatoriamente una muestra de cada uno de los estratos, y se quita esta muestra del conjunto original. Dado que al dividir en estratos habrá estratos vacíos (sin muestras), este proceso se repite iterativamente con las muestras restantes del conjunto original hasta obtener el número de muestras deseadas.

- **Muestreo basado en la curva gamma**: Este muestreo extrae directamente las muestras del conjunto original de acuerdo a una curva gamma. A diferencia del anterior, no se produce ninguna división según umbrales, simplemente se muestrea según la curva gamma y un paso que se calcula en función del número de muestras que se deseen.

En la Figura 6.13 pueden verse los resultados de la reconstrucción del vídeo *Coke* para cada uno de los métodos, midiendo su calidad con PSNR. Se incluye también en la figura, para cada uno de los métodos, un histograma de las varianzas de las muestras escogidas, esto es, la distribución de varianzas de los bloques seleccionados para el entrenamiento. Los mejores resultados se dan para selección aleatoria y selección por varianza, siendo los resultados para el resto de métodos de selección de calidad considerablemente menor. Se pensó que al coger un mayor número de muestras de alta varianza (esto es, con las selecciones mediante función gamma) se obtendría una mejor reconstrucción de los detalles pero, tal y como se observa en la Figura 6.13 los mejores resultados se obtienen cuando la distribución de las varianzas de las muestras escogidas para el entrenamiento se asemeja más a la distribución de muestras los vídeos originales, es decir, cuanto más se parecen sus histogramas. El método de selección de varianza ofrece mejores resultados que el aleatorio porque ofrece el mejor compromiso entre evitar un exceso de bloques uniformes y respetar la distribución original. Además, este método conlleva un menor tiempo de cálculo, como puede verse en la tabla 6.2. Por todas estas razones, el método de selección por varianza es el escogido como método por defecto.

6.5. Algoritmo de reconstrucción

Recordemos que el algoritmo de reconstrucción es el algoritmo utilizado para determinar los coeficientes α que multiplican a cada uno de los elementos del diccionario para obtener la señal (o vídeo), resolviendo la Ecuación 4.3. Además, el algoritmo de reconstrucción se ha de utilizar tanto en el entrenamiento—porque la selección de los átomos o elementos del diccionario mediante K-SVD implica la resolución del mencionado problema de minimización L1—, como

[1]La varianza se normaliza con respecto a su máximo para obtener valores entre 0 y 1 y poder muestrear con la curva gamma que toma valores entre 0 y 1.

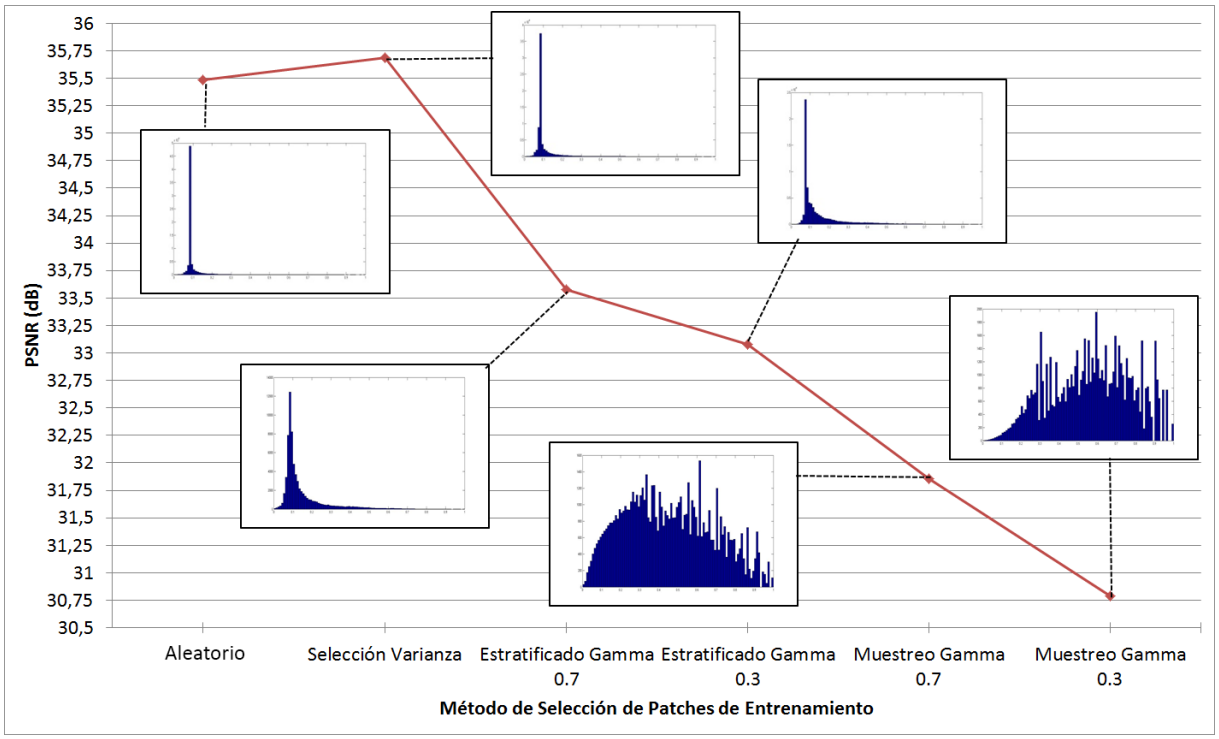

Figura 6.13: PSNR en función del método de selección utilizado para escoger los bloques de entrenamiento.

en la reconstrucción del vídeo final a partir del diccionario entrenado y la señal muestreada. Se analizan en este trabajo dos de los principales algoritmos de resolución de dicha minimización L1, OMP y Lasso (explicados en el Anexo C), y su utilidad para el entrenamiento y para la reconstrucción final.

En la Figura 6.14 se exploran varias combinaciones de estos dos algoritmos para determinar cuál es la mejor combinación de ambos, mostrando el PSNR de cada uno de los seis vídeos reconstruidos utilizando OMP o Lasso para la reconstrucción y/o para el entrenamiento de los diccionarios. Los tiempos medios correspondientes pueden verse en la Tabla 6.3. Por lo general, las mejores reconstrucciones se producen cuando Lasso es el algoritmo utilizado para la reconstrucción, siendo menor la influencia del algoritmo utilizado para el entrenamiento. A esto se añade el factor de que, dado que Lasso es paralelizable, el entrenamiento con este algoritmo es

Método	Tiempo(segundos)
Aleatorio	868,9419
Selección varianza	781,986
Estratificado Gamma 0,7	1295,614
Estratificado Gamma 0,3	971,2521
Gamma 0,7	1001,485
Gamma 0,3	976,7386

Tabla 6.2: Tabla de tiempos medios de reconstrucción según el método de selección de bloques de entrenamiento del diccionario.

	Lasso/Lasso	OMP/OMP	OMP/Lasso	Lasso/OMP
Tiempo(segundos)	9185	8545	8156	10347

Tabla 6.3: Tiempos medios de las reconstrucciones para diferentes combinaciones de algoritmos de Entrenamiento/Reconstrucción.

más rápido, 432000 segundos con OMP y 212220 con Lasso. El vídeo *Spring* vuelve a presentar un comportamiento claramente diferente al resto, que achacamos a sus particulares características espacio-temporales (Sección 6.2).

Por tanto, decidimos utilizar Lasso tanto para las reconstrucciones como para el entrenamiento.

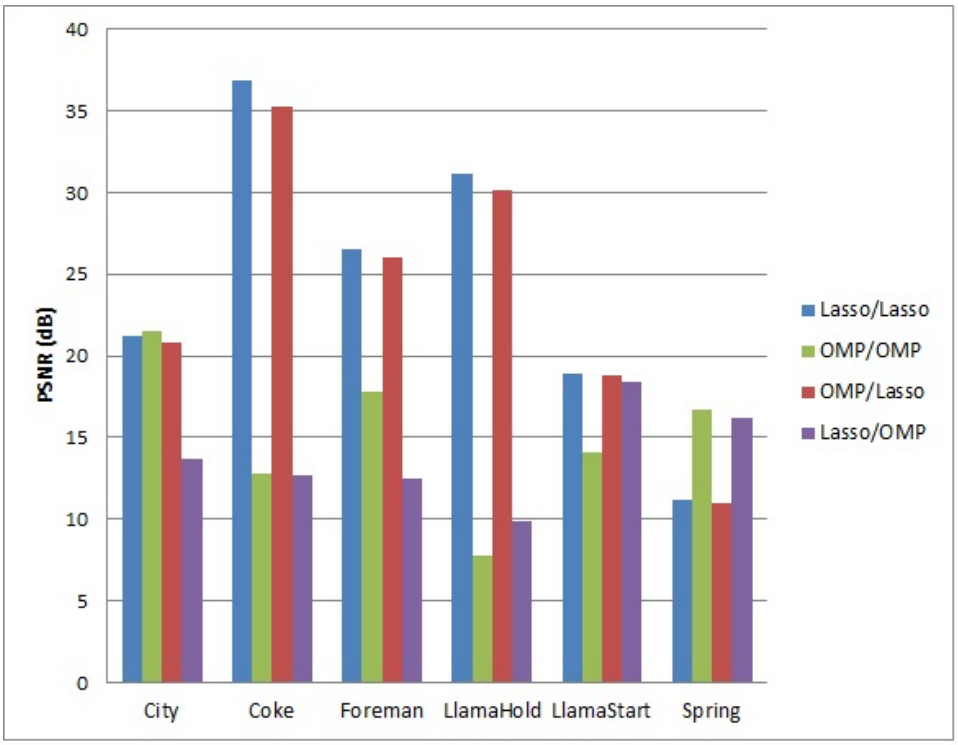

Figura 6.14: Comparación del PSNR de los resultados para diferentes vídeos y diferentes combinaciones de algoritmos de Entrenamiento/Reconstrucción.

6.6. Matriz de medida: función del obturador

La elección de una matriz de medida adecuada es determinante para los resultados de un esquema de *compressive sensing*. Como se ha explicado en la Sección 4.3, debe cumplir una serie de propiedades matemáticas (incoherencia, RIP), pero al mismo tiempo ser físicamente realizable, e implementable mediante un obturador y/o una máscara o LCoS (panel de cristal líquido cuyos píxeles pueden regularse para dejar pasar o bloquear la luz, formando una máscara dinámica). En esta sección se comparan diferentes matrices de medida físicamente implementables, que

han sido descritas previamente en la Sección 4.3.2. Se muestran y analizan los resultados de esta comparación para el vídeo *Coke*; la diferencia de resultados en las reconstrucciones es tan grande que las conclusiones para un solo vídeo pueden generalizarse para el resto.

En la Figura 6.15 puede verse el PSNR de las reconstrucciones obtenidas así como un fotograma representativo del vídeo reconstruido para cada una de las matrices de medida (o funciones del obturador). Los mejores resultados se obtienen para la función *pixel-wise*. Esto se debe, principalmente, a que garantiza que para cada fotograma al menos uno de los píxeles de cada bloque en los que se divide el vídeo para su reconstrucción será muestreado, y además es un muestreo más aleatorio que otros de los empleados. El obturador *global*, de muy sencilla implementación, ofrece resultados demasiado borrosos, en los que no se pueden recuperar las altas frecuencias, que se han eliminado irreversiblemente. Los obturadores basados en *rolling shutter* (*rolling* o *coded rolling*), interesantes porque muchas cámaras utilizan este método para controlar el obturador, simplificando la necesidad de modificar el hardware, ofrecen los resultados de menor calidad, con visibles artefactos. Mientras, el *flutter-shutter* ofrece unos resultados de calidad media que no compensan el coste de su implementación. Esto se debe a que, al muestrear tan solo algunos fotogramas de manera aleatoria, hay fotogramas completos que se pierden por lo que la calidad en la reconstrucción es muy inferior a la de la función *pixel-wise*, que es la finalmente elegida.

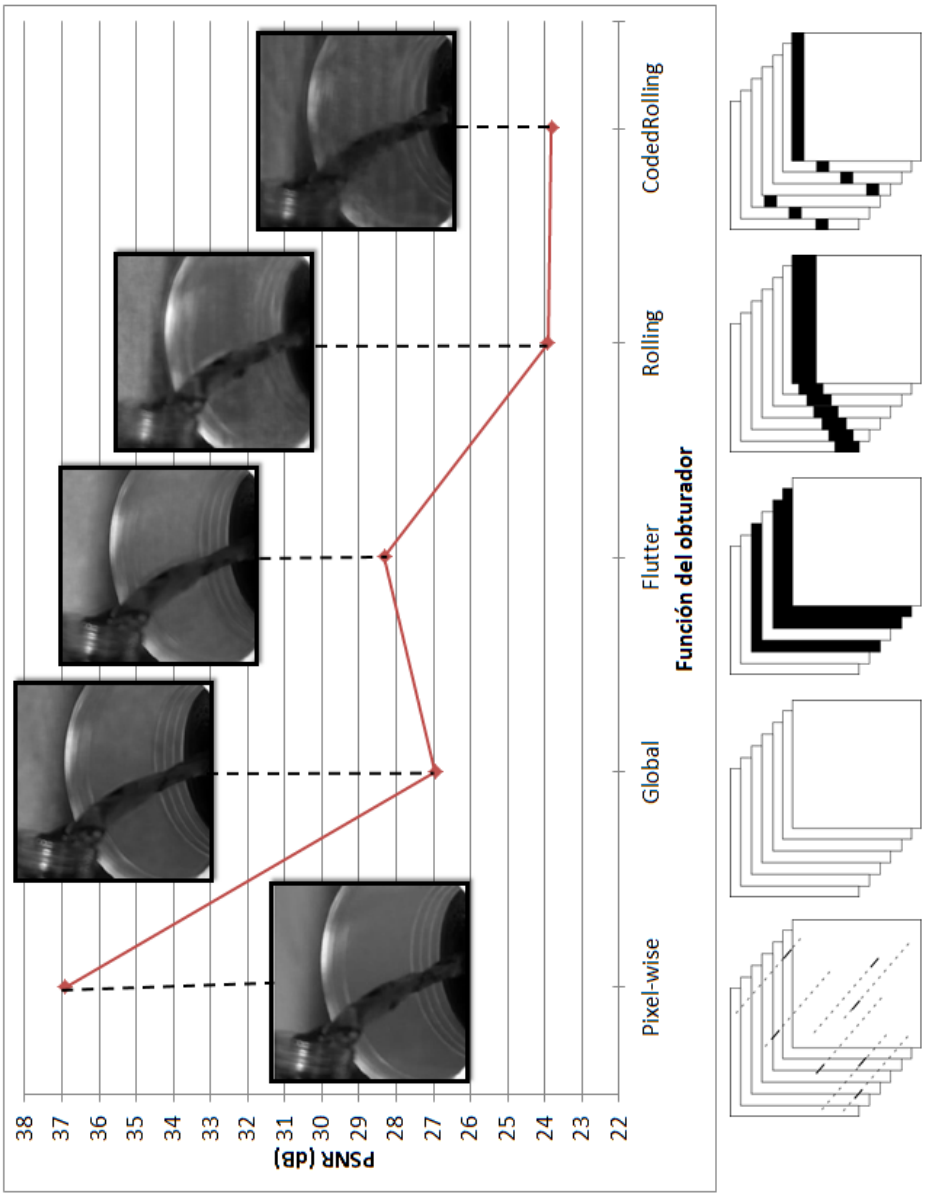

Figura 6.15: Comparación del PSNR de los resultados para diferentes tipos de funciones del obturador

7. Resultados

A lo largo del Capítulo 6 se han analizado diversos parámetros tanto de captura como de reconstrucción. Tal y como se ha justificado, los mejores parámetros en términos de resultados y tiempos de computación son los siguientes.

- Diccionario: Entrenado con algoritmo Lasso, tamaño del átomo de $7 \times 7 \times 20$, muestras de entrenamiento escogidas mediante selección de varianza.
- Matriz de medida: Función del obturador *pixel-wise* aleatorio.
- Reconstrucción: Algoritmo Lasso.

Como ejemplo, para estos parámetros se ha reconstruido el vídeo de la llama prendiéndose en el mechero completo. Se ha escogido este vídeo porque, tal y como se ha visto en el Capítulo 6 con los ejemplos *LlamaHold* y *LlamaStart*, contiene tanto partes con cambios más marcados y rápidos que se reconstruyen peor (al prenderse la llama), como partes con cambios leves que se reconstruyen mejor (cuando la llama esta estabilizada). Por ello, y debido a la limitación de recursos que han impedido una reconstrucción completa de todos los vídeos, se ha considerado este vídeo como ejemplo representativo.

Este vídeo consta de 200 fotogramas y una resolución de 512×512 píxeles y esta dividido en 10 fragmentos de 20 fotogramas cada uno. Las imágenes codificadas correspondientes a estos 10 fragmentos pueden verse en la Figura 7.1. Se muestran también dos ejemplos del vídeo reconstruido, en cada uno aparecen los 20 fotogramas correspondientes a la reconstrucción de dos fragmentos diferentes, uno de ellos al prenderse la llama, en la Figura 7.2 y el otro cuando la llama ya esta estabilizada, en la Figura 7.3. Asimismo, el tiempo medio de reconstrucción son 82851 segundos para cada fragmento de 20 fotogramas.

Figura 7.1: Imágenes codificadas para el vídeo Llama. Orden temporal de izquierda a derecha y de arriba a abajo. Cada imagen codificada contiene información para la reconstrucción de 20 fotogramas de vídeo. Las imágenes con marcos de color muestran sus reconstrucciones en la Figura 7.1 (Azul) y en la Figura 7.3 (amarillo).

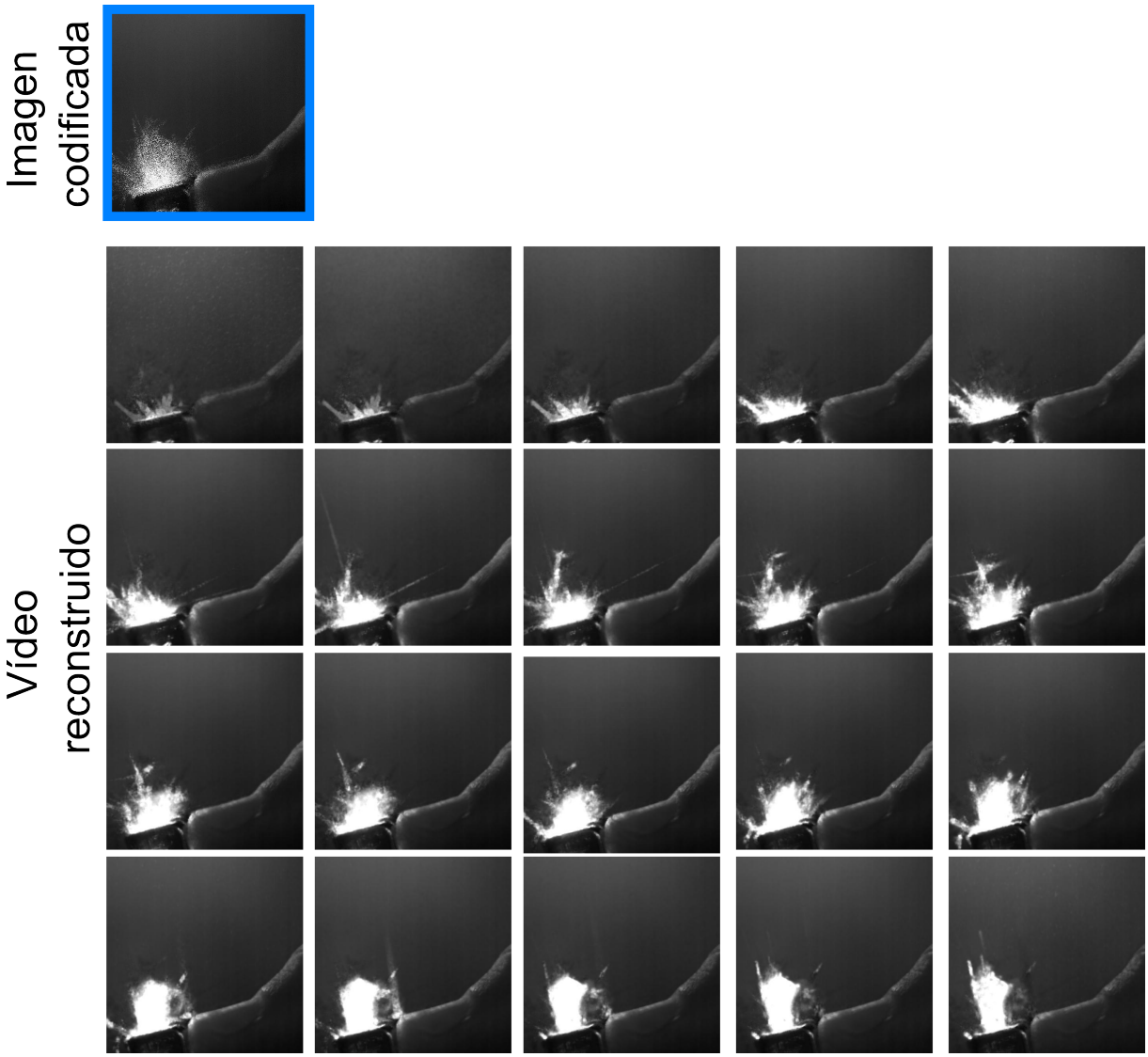

Figura 7.2: Fotogramas reconstruidos para la llama prendiéndose. En la imagen se pueden observar los 20 fotogramas reconstruidos a partir de la imagen codificada. Orden temporal de izquierda a derecha y de arriba a abajo.

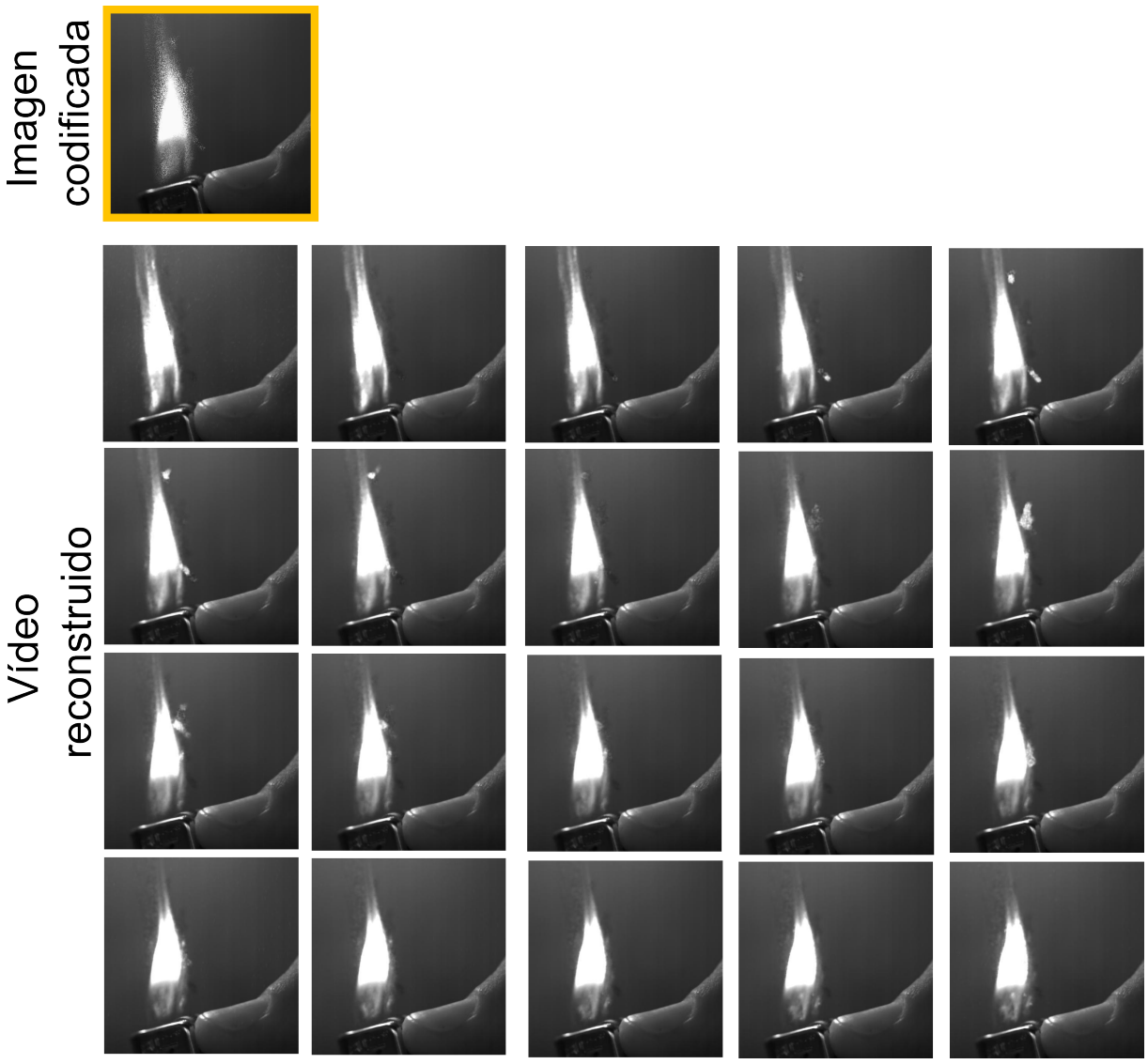

Figura 7.3: Fotogramas reconstruidos para la llama estabilizada. En la imagen se pueden observar los 20 fotogramas reconstruidos a partir de la imagen codificada. Orden temporal de izquierda a derecha y de arriba a abajo.

8. Conclusiones y trabajo futuro

En este proyecto hemos presentado un sistema de de captura y reconstrucción de vídeo de alta resolución temporal, a partir de una sola imagen codificada. Para ello hemos utilizado técnicas de *compressive sensing*. El desarrollo básico de la técnica utilizada se ha basado en el trabajo de Hitomi et al. [8, 9], que posteriormente hemos analizado y sobre el que hemos añadido ciertas mejoras. En concreto, hemos analizado una serie de escenas (grabadas en vídeos originales de alta resolución) y hemos encontrado una métrica para caracterizarlas, que nos permite explicar las diferencias de resultados en su reconstrucción con unos mismos parámetros. También hemos explorado el entrenamiento de diccionarios con el algoritmo *K-SVD*, analizando los resultados obtenidos en la reconstrucción en función del tamaño de los átomos, y comparando los resultados de este algoritmo frente a un diccionario ya existente, en este caso DCT. Hemos profundizado en el método de selección de muestras de entrenamiento de entre un conjunto, hallando el método de *selección de varianza* que arroja los mejores resultados en las reconstrucciones. Asimismo, hemos comparado el rendimiento de dos algoritmos de reconstrucción, Lasso y OMP, llegando a la conclusión de que con el primero, en general, se obtiene una mejor calidad de los resultados así como una mayor velocidad en el cálculo de los coeficientes. Por otra parte, se ha analizado la compatibilidad de diversos obturadores comúnmente utilizados con la captura de vídeo mediante *compressive sensing*, corroborando formalmente que ninguno de ellos es superior al propuesto originalmente por Liu et al. en [9].

Todo esto proporciona unas serie de contribuciones al campo que pretendemos completar y consolidar en una sumisión de un artículo científico. Sin embargo, al ser éste es un campo relativamente novedoso, nuestra exploración del espacio de parámetros y el análisis de su rendimiento están lejos de ser completos, abriendo muchas posibles vías de trabajo futuro. Un camino a seguir sería continuar este trabajo con un análisis más exhaustivo, que quizá lleve a combinaciones de parámetros óptimas según el tipo de escena a capturar.

Por otra parte, en el campo del entrenamiento de diccionarios, el algoritmo *K-SVD* que es el usado en este proyecto, y en gran cantidad de trabajos, es un algoritmo *genérico*. Una mejora en la reconstrucción se puede buscar con el desarrollo de un algoritmo de entrenamiento perceptual, esto es, incorporar al entrenamiento modelos computacionales de visión o métricas perceptuales, resultando reconstrucciones mejores a vista del ojo humano. Este tipo de métricas ya se han usado anteriormente, por ejemplo, Masia et al. [3, 44] hacen uso de aperturas codificadas optimizadas con métricas perceptuales para corrección del desenfoque.

Finalmente, los sensores de imagen y el *hardware* de las cámaras están en continua evolución, por lo que es de esperar que en algún momento algunas de las limitaciones actuales del sistema de captura se vayan eliminando. Esto podría estar asociado con la construcción de obturadores

más cercanos a las matrices de medida teóricas, que sean capaces de capturar la señal de una manera más óptima para su reconstrucción. Aunque en este proyecto nos hemos limitado a trabajar en simulación, se podría considerar el uso de pantallas de cristal líquido sobre silicona (LCoS), que actúan sobre la exposición a nivel de píxel de manera independiente, ofreciendo gran versatilidad.

A nivel personal, el desarrollo de este proyecto ha sido una experiencia decisiva para mí, ya que he podido profundizar en un campo que siempre me ha gustado, el procesado de imagen, desde una perspectiva nueva y diferente. Además, gracias a este proyecto y la buena acogida del grupo de investigación en el que lo he desarrollado, he descubierto en la investigación un camino a seguir, algo que nunca había considerado, y voy a comenzar el doctorado, bajo la tutela de mi directora y el ponente de este Proyecto Fin de Carrera.

Bibliografía

[1] Michael B. Wakin, Jason N. Laska, Marco F. Duarte, Dror Baron, Shriram Sarvotham, Dharmpal Takhar, Kevin F. Kelly, and Richard G. Baraniuk. Compressive imaging for video representation and coding. In *In Proceedings of Picture Coding Symposium (PCS*, 2006.

[2] J. Gu, Y. Hitomi, T. Mitsunaga, and S.K. Nayar. Coded Rolling Shutter Photography: Flexible Space-Time Sampling. In *IEEE International Conference on Computational Photography (ICCP)*, Mar 2010.

[3] Belen Masia, Adrian Corrales, Lara Presa, and Diego Gutierrez. Coded apertures for defocus deblurring. In *SIACG*, 2011.

[4] Justin Romberg. L1-magic. Accedido por Ã°ltima vez el 31 de Agosto de 2014.

[5] M. Aharon, M. Elad, and A. Bruckstein. Svdd: An algorithm for designing overcomplete dictionaries for sparse representation. *Trans. Sig. Proc.*, 54(11):4311–4322, November 2006.

[6] S. Kleinfelder, SukHwan Lim, Xinqiao Liu, and A El Gamal. A 10000 framesps cmos digital pixel sensor. *Solid-State Circuits, IEEE Journal of*, 36(12):2049–2059, Dec 2001.

[7] Gregory K. Wallace. The jpeg still picture compression standard. *Communications of the ACM*, pages 30–44, 1991.

[8] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S.K. Nayar. Video from a Single Coded Exposure Photograph using a Learned Over-Complete Dictionary. In *IEEE International Conference on Computer Vision (ICCV)*, Nov 2011.

[9] D. Liu, J. Gu, Y. Hitomi, M. Gupta, T. Mitsunaga, and S.K. Nayar. Efficient Space-Time Sampling with Pixel-wise Coded Exposure for High Speed Imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99:1, 2013.

[10] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[11] Ronald A. DeVore. Deterministic constructions of compressed sensing matrices. *J. Complex.*, 23(4-6):918–925, August 2007.

[12] Holger Rauhut, Karin Schnass, and Pierre Vandergheynst. Compressed sensing and redundant dictionaries. *CoRR*, abs/math/0701131, 2007.

[13] Jian Wang and Byonghyo Shim. On the recovery limit of sparse signals using orthogonal matching pursuit. *IEEE Transactions on Signal Processing*, 60(9):4973–4976, 2012.

[14] Vladimir N. Temlyakov and Pavel Zheltov. On performance of greedy algorithms. *Journal of Approximation Theory*, 163(9):1134–1145, 2011.

[15] Shamgar Gurevich and Ronny Hadani. Incoherent dictionaries and the statistical restricted isometry property. *CoRR*, abs/0809.1687, 2008.

[16] Florian M. Sebert, Leslie Ying, and Yi Ming Zou. Toeplitz block matrices in compressed sensing. *CoRR*, abs/0803.0755, 2008.

[17] J. Provost and F. Lesage. The application of compressed sensing for photo-acoustic tomography. *Medical Imaging, IEEE Transactions on*, 28(4):585–594, April 2009.

[18] Michael Lustig, David L. Donoho, Juan M. Santos, and John M. Pauly. Compressed sensing mri. In *IEEE SIGNAL PROCESSING MAGAZINE*, 2007.

[19] W.U. Bajwa, A Sayeed, and R. Nowak. Compressed sensing of wireless channels in time, frequency, and space. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 2048–2052, Oct 2008.

[20] W.U. Bajwa, J. Haupt, AM. Sayeed, and R. Nowak. Compressed channel sensing: A new approach to estimating sparse multipath channels. *Proceedings of the IEEE*, 98(6):1058–1076, June 2010.

[21] K. Marwah, G. Wetzstein, Y. Bando, and R. Raskar. Compressive Light Field Photography using Overcomplete Dictionaries and Optimized Projections. *ACM Trans. Graph.*, 32(4):1–11, 2013.

[22] Yebin Liu et al. Spatial-spectral Encoded Compressive Hyperspectral Imaging. *ACM Trans. Graph. (conditionally accepted to SIGGRAPH Asia)*, 33(6), 2014.

[23] Atsushi Ito, Salil Tambe, Kaushik Mitra, Aswin Sankaranarayanan, and Ashok Veeraraghavan. Compressive epsilon photography for post-capture control in digital imaging. 2014.

[24] Ankit Gupta, Pravin Bhat, Mira Dontcheva, Brian Curless, Oliver Deussen, and Michael Cohen. Enhancing and experiencing spacetime resolution with videos and stills. In *International Conference on Computational Photography*. IEEE, 2009.

[25] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Marc Levoy, and Mark Horowitz. High-speed videography using a dense camera array. In *CVPR (2)*, pages 294–301, 2004.

[26] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: Motion deblurring using fluttered shutter. *ACM Trans. Graph.*, 25(3):795–804, July 2006.

[27] Richard Baraniuk, Mark Davenport, Ronald Devore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constr. Approx*, 2008, 2007.

[28] Dongdong Ge, Xiaoye Jiang, and Yinyu Ye. A note on complexity of lp minimization, 2010.

[29] Scott Shaobing Chen, David L. Donoho, Michael, and A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1998.

[30] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, Feb 2006.

[31] M.AT. Figueiredo, R.D. Nowak, and S.J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586–597, Dec 2007.

[32] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[33] Deanna Needell and Roman Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit, submitted. Technical report, 2007.

[34] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. Technical report, California Institute of Technology, Pasadena, 2008.

[35] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean luc Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. Technical report, 2006.

[36] M. A. Iwen. Combinatorial sublinear-time fourier algorithms. *Found. Comput. Math.*, 10(3):303–338, June 2010.

[37] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. One sketch for all: Fast algorithms for compressed sensing. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 237–246, New York, NY, USA, 2007. ACM.

[38] Gitta Kutyniok. Compressed sensing: Theory and applications. *CoRR*, abs/1203.3815, 2012.

[39] Photron sa2 camera datasheet. `http://www.photron.com/datasheet/FASTCAM_SA2.pdf`. Accedido por Ã°ltima vez el 31 de Agosto de 2014.

[40] Q. Huynh-Thu and M. Ghanbari. Scope of validity of psnr in image/video quality assessment. 44:800–801, 2008.

[41] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multi-scale structural similarity for image quality assessment. In *in Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers, (Asilomar*, pages 1398–1402, 2003.

[42] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, 1895.

[43] John H. McDonald. Spearman rank correlation. `http://www.biostathandbook.com/spearman.html`. Accedido por Ã°ltima vez el 31 de Agosto de 2014.

[44] Belen Masia, Lara Presa, Adrian Corrales, and Diego Gutierrez. Perceptually-optimized coded apertures for defocus deblurring. *Computer Graphics Forum*, 31(6), 2012.

[45] Emmanuel J. CandÃ¨s and Terence Tao. Near optimal signal recovery from random projections: Universal encoding strategies?, 2006.

[46] Dan Kalman. A singularly valuable decomposition: The svd of a matrix. *College Math Journal*, 27:2–23, 1996.

[47] P. Berkes, R. E. Turner, and M. Sahani. On sparsity and overcompleteness in image models. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.

[48] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, Angela Y. Wu, Senior Member, and Senior Member. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:881–892, 2002.

[49] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[50] Y. C. Pati, R. Rezaiifar, Y. C. Pati R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*, pages 40–44, 1993.

Anexo A. Propiedades de la matriz de medida

A.1. Incoherencia

Se entiende por incoherencia entre dos matrices, ϕ y ψ que ningún elemento de la matriz ϕ puede ser obtenido mediante combinación lineal de los elementos de ψ y viceversa. La coherencia entre estas dos matrices viene dada por la ecuación A.1

$$\mu(\phi, \psi) = \sqrt{n} max_{1 \leq k,j \leq n} | < \phi_k, \psi_j > | \tag{A.1}$$

con $\mu(\phi, \psi) \in [1, \sqrt{n}]$. Esta ecuación mide la coherencia máxima para cada par de elementos de ϕ y ψ, es decir, la correlación máxima entre cualquier par de ϕ y ψ. Un valor bajo de coherencia indica que una señal *sparse* en una de las bases, tendrá una representación densa en la otra, que es el objetivo que se busca en *compressive sensing*. Para $\mu(\phi, \psi) = 1$ se dice que las bases son *maximalmente incoherentes*.

Para matrices de medida ϕ con una distribución aleatoria Gaussiana o de Bernouille con muestras independientes e identicamente distribuidas se tiene una incoherencia alta con respecto a cualquier base fija ortonormal ψ, tal y como prueban Candes y Tao en [45]. Para estos casos, la coherencia entre ϕ y ψ será con alta probabilidad $\sqrt{2 \log n}$.

A.2. RIP: Restricted Isometry Property

En el trabajo de Candes y Wakin en [27] se define la constante isométrica δ_S de una matriz A para cada entero $S = 1, 2, \ldots$ como el número más pequeño tal que

$$(1 - \delta_S) \parallel x \parallel_2^2 \leq \parallel Ax \parallel_2^2 \leq (1 + \delta_S) \parallel x \parallel_2^2 \tag{A.2}$$

Para todo vector x *S-sparse*.

Se dice que la matriz A obedece la propiedad *RIP* de orden S si δ_S no está demasiado cerca de uno. Una descripción equivalente es decir que todos los sub-conjuntos de S columnas de A son *casi* ortogonales (las columnas de A no pueden ser exactamente ortogonales ya que se tienen más columnas que filas). En este mismo trabajo [27] se puede encontrar la demostración de porqué las matrices aleatorias cumplen esta propiedad y son adecuadas para el muestreo mediante *compressive sensing*.

Anexo B. Entrenamiento de diccionarios: Algoritmo K-SVD

B.1. Introducción

En este anexo se desarrolla el funcionamiento del algoritmo K-SVD propuesto por Aharon, Elad y Bruckstein en [5]. K-SVD es un algoritmo de aprendizaje de diccionarios *overcomplete* especialmente desarrollado para representaciones *sparse* haciendo uso de la descomposición en valores singulares o *SVD* [46]. *Overcompleteness* es un término que denomina a un sistema completo al que si le quitamos un vector, resulta en un sistema completo [47]. Este algoritmo es una generalización del algoritmo *K-Means* [48] que consiste en dividir N observaciones en K *clusters* o núcleos en los que cada observación pertenece al *cluster* con la media más cercana. K-SVD alterna iterativamente entre codificar los datos de entrenamiento de manera *sparse* con el diccionario actual y actualizar los átomos de dicho diccionario para que se ajusten mejor a los datos de entrenamiento.

El objetivo del algoritmo es aprender un diccionario $D \in \Re^n \times K$ con K átomos tal que la señal $X \in \Re^n$ pueda ser representada de manera *sparse* como combinación lineal de estos átomos. Para ello el vector de coeficientes αdebe satisfacer al menos que $X \approx D\alpha$ de manera que $\parallel X - D\alpha \parallel_p \leq \epsilon$ para algún valor pequeño de ϵ y alguna norma L_p. Típicamente la norma se elige como L_1, L_2, o L_∞.

Si $n < K$ y D es una matriz de rango completo ($rango(D) = n$), se tienen un infinito número de soluciones para el problema de representación, por lo que se deben poner restricciones en la solución. Para asegurar un resultado *sparse* debe imponerse que

$$\min_{\alpha} \|\alpha\|_0 \text{ sujeto a} X = D\alpha \tag{B.1}$$

o

$$\min_{\alpha} \|\alpha\|_0 \text{ sujeto a } \parallel X - D\alpha \parallel_2 \leq \epsilon \tag{B.2}$$

donde la norma L_0 cuenta el número de elementos distintos de cero del vector α. Debido a la complejidad del problema con la norma L_0, se consideran soluciones aproximadas para el cálculo de los coeficientes α, para saber más acerca de estos algoritmos de aproximación consultar el Anexo C.

B.2. Algoritmo K-SVD

El objetivo es encontrar el conjunto de átomos que represente de la mejor manera posible las muestras $X_i{}_{i=1}^{N}$, es decir, que minimice el error, bajo el supuesto de vecino mas cercano resolviendo

$$\min_{D,\alpha}\{\| X - D\alpha \|_F^2\}\text{sujeto a}\forall i, \| \alpha_i \|_0 \leq T_0 \tag{B.3}$$

o, escrito de otra manera

$$\min_{D,\alpha} \sum_i \| \alpha_i \|_0 \text{ sujeto a}\forall i, \| X - D\alpha \|_F^2 \leq \epsilon \tag{B.4}$$

con $\| A \|_F$ la norma Frobenius definida como $\| A \|_F = \sqrt{\sum_{i,j} A_{ij}^2}$ y siendo T_0 el número de elementos distintos de cero del vector de coeficientes α.

En el algoritmo K-SVD primero debe fijarse un diccionario inicial D y un conjunto de datos de entrenamiento X. El primer paso consiste en codificar de manera *sparse* el conjunto de datos de entrenamiento en el diccionario D mediante un algoritmo de aproximación, es decir, obtener el vector de coeficientes α. Después, el siguiente paso es buscar una mejora del diccionario D. Sin embargo, no se puede encontrar un mejor diccionario completo de una vez, por lo que el proceso consiste en actualizar solo una columna del diccionario cada vez que se fija un nuevo vector α.

Se asume pues que α y D están fijados y debe actualizarse la columna d_k y los coeficientes que le corresponden, la $k-esima$ fila en α. Entonces, el término de penalización puede escribirse como

$$\| X - D\alpha \|_F^2 = | X - \sum_{j=1}^{K} d_j \alpha_T^j |_F^2 = | (X - \sum_{j\neq k} d_j \alpha_T^j) - d_k \alpha_T^k) |_F^2 = \| E_k - d_k \alpha_T^k \|_F^2 \tag{B.5}$$

donde α_T^k denota la $k-esima$ fila de α y la matriz E_k es el error para todas las muestras cuando el $k-esimo$ átomo se elimina.

Se ha descompuesto la multiplicación $D\alpha$ en suma de K matrices de rango 1. De ellas, se consideran $K-1$ términos fijados y el $k-esimo$ permanece desconocido. Después de este paso, se podría aplicar SVD para encontrar la matriz de rango 1 que aproxima E_k y esto minimizaría el error tal y como se ha definido, sin embargo, la condición de solución *sparse* no esta aplicada todavía por lo que no sería el resultado deseado. Para tener en cuenta esta condición, se define ω_k como el grupo de índices apuntando a las muestras X_i que usan el átomo d_k, i.e, aquellas donde $\alpha_T^k(i)$ es distinto de cero. Entonces

$$\omega_k = \{i \mid 1 \leq i \leq K, \alpha_T^k(i) \neq 0\} \tag{B.6}$$

Después, se define Ω_k como una matriz de tamaño $N \times \mid \omega_k \mid$, con unos en las entradas correspondientes a $(\omega_k(i), i)$ y ceros en el resto. Al multiplicar $\alpha_R^k = \alpha_T^k \Omega_k$, esto reduce el vector fila α_T^k descartando las entradas iguales a cero, resultando el vector fila α_R^k de longitud $\mid \omega_k \mid$. De igual manera, la multiplicación $X_k^R = X\Omega_k$ es el sub-conjunto de muestras que está usando

actualmente el átomo d_k, y la multiplicación $E_k^R = E_k\Omega_k$ son las columnas error correspondientes a las muestras que usan el átomo d_k. Así pues, el problema de minimización se convierte en

$$\| E_k\Omega_k - d_k\alpha_T^k\omega_k \|_F^2 = \| E_k^R - d_k\alpha_R^k \|_F^2 \tag{B.7}$$

y puede resolverse directamente via SVD, que descompone la matriz E_k^R como $E_k^R = U \triangle V^T$. Se define la solución para d_k como la primera columna de U y el vector de coeficientes α_R^k como la primera columna de V multiplicada por $\triangle\ (1,1)$. Es importante remarcar que las columnas de D deben permanecer normalizadas.

Después de actualizar el diccionario completo, el proceso vuelve a calcular α y a resolver de nuevo D iterativamente. El proceso resumido del algoritmo puede verse en la Figura B.1

Task: Find the best dictionary to represent the data samples $\{\mathbf{y}_i\}_{i=1}^N$ as sparse compositions, by solving

$$\min_{\mathbf{D},\mathbf{X}} \left\{\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2\right\} \quad \text{subject to} \quad \forall i,\ \|\mathbf{x}_i\|_0 \le T_0.$$

Initialization : Set the dictionary matrix $\mathbf{D}^{(0)} \in \mathbb{R}^{n\times K}$ with ℓ^2 normalized columns. Set $J = 1$.
Repeat until convergence (stopping rule):

- *Sparse Coding Stage*: Use any pursuit algorithm to compute the representation vectors $\mathbf{x}_i$ for each example $\mathbf{y}_i$, by approximating the solution of

$$i = 1, 2, \ldots,\ N, \quad \min_{\mathbf{x}_i} \left\{\|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2\right\} \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 \le T_0.$$

- *Codebook Update Stage*: For each column $k = 1, 2, \ldots, K$ in $\mathbf{D}^{(J-1)}$, update it by
  - Define the group of examples that use this atom, $\omega_k = \{i |\ 1 \le i \le N,\ \mathbf{x}_T^k(i) \ne 0\}$.
  - Compute the overall representation error matrix, $\mathbf{E}_k$, by

$$\mathbf{E}_k = \mathbf{Y} - \sum_{j\ne k} \mathbf{d}_j\mathbf{x}_T^j.$$

  - Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to ω_k, and obtain $\mathbf{E}_k^R$.
  - Apply SVD decomposition $\mathbf{E}_k^R = \mathbf{U}\mathbf{\Delta}\mathbf{V}^T$. Choose the updated dictionary column $\tilde{\mathbf{d}}_k$ to be the first column of $\mathbf{U}$. Update the coefficient vector $\mathbf{x}_R^k$ to be the first column of $\mathbf{V}$ multiplied by $\mathbf{\Delta}(1,1)$.
- Set $J = J + 1$.

Figura B.1: Proceso iterativo del algoritmo K-SVD [5]

Anexo C. Algoritmos OMP y LASSO

En este anexo se presentan dos algoritmos para resolver el problema de minimización que presenta *compressive sensing* a la hora de representar una señal en una base de manera *sparse* tal y como se indica en la siguiente ecuación

$$\min_{\alpha} \|\alpha\|_0 \text{ sujeto a } Y = \Theta\alpha \tag{C.1}$$

Para ello, a lo largo del tiempo se han estudiado diferentes tipos de enfoques, entre ellos, optimización convexa, algoritmos voraces y algoritmos combinatorios. Los algoritmos por optimización convexa requieren menos muestras pero son más complejos computacionalmente. En el otro extremo, los algoritmos combinatorios son muy rápidos pero necesitan muchas medidas, lo cual esta en contraposición con el objetivo de *compressive sensing*. Por otra parte, los algoritmos voraces son un buen compromiso entre número de medidas y complejidad computacional. En este anexo se presentan los dos algoritmos usados durante el proyecto, un algoritmo de optimización convexa: LASSO, y un algoritmo voraz: OMP.

C.1. LASSO: Least Absolute Shrinkage and Selection Operator

LASSO es un algoritmo de optimización convexa. En general, para estos algoritmos, si se cumplen las condiciones de señal *sparse* y las matrices ϕ y ψ (la matriz de medida y la base que conforman la matriz Θ) cumplen las condiciones de incoherencia y la propiedad de isometría restrictiva tal y como se explican en el Anexo A, el problema puede aproximarse mediante la norma L_1 de manera

$$\min_{\alpha} \|\alpha\|_1 \text{ sujeto a } Y = \Theta\alpha \tag{C.2}$$

Por otra parte, si las medidas se ven afectadas por ruido, se necesita definir una constante de error, quedando la ecuación

$$\min_{\alpha} \|\alpha\|_1 \text{ sujeto a } \| \Theta\alpha - Y \|_2^2 \leq \epsilon \tag{C.3}$$

para un $\epsilon > 0$. En el algoritmo LASSO, se utiliza un factor de regularización, entonces el problema es equivalente a la versión sin restricciones dada por

$$\min_{\alpha} \frac{1}{2} \| \Theta\alpha - Y \|_2^2 + \lambda \| \alpha \|_1 \tag{C.4}$$

Para resolver este problema, se plantea una modificación del algoritmo LAR(Least Angle Regression) llamada LARS(Least angle regression and shrinkage) que es capaz de resolver la regularización planteada por LASSO. En este anexo se describe una visión general del algoritmo, para una descripción más detallada consultar el trabajo de Efron et al. en [49].

Siendo, y la señal a representar, A la matriz que define la base del dominio en el que se quiere representar y, y x los coeficientes a calcular de esta representación, el algoritmo LAR consiste en

- Establecer todos los $x_j = 0$.
- Encontrar el predictor A_j más correado con y.
- Aumentar el coeficiente x_j en la dirección del signo de su correlación con y.
- calcular el residuo $r = y - Ax$
- Repetir hasta que otro predictor A_k tenga tanta correlación con el residuo actual como A_j y añadirlo al conjunto de predictores ativos.
- Aumentar (x_j, x_k) en su dirección conjunta por mínimos cuadrados hasta que otro predictor A_m tenga tanta correlación con el residuo actual como el conjunto de predictores activos.
- Continuar hasta que todos los predictores estén en el modelo

La modificación necesaria para adaptarlo a LASSO consiste en: si un coefficiente x_j diferente de cero llega a cero, eliminarlo del conjunto activo y recalcular la dirección conjunta.

C.2. OMP: Orthogonal Matching Pursuit

Los algoritmos voraces aproximan iterativamente los coeficientes y el soporte de la señal original. Tienen la ventaja de ser muy rápidos y fáciles de implementar. Uno de los más conocidos es *OMP* [50] que se describe a continuación.

Siendo, y la señal a representar, A la matriz que define la base del dominio en el que se quiere representar y, y x los coeficientes a calcular de esta representación.

Entrada

- Matriz $A = (a_i)_{i=1}^{n} \in \Re^{m \times n}$ y el vector $x \in \Re^n$
- Umbral del error ϵ

Algoritmo

- Establecer $k = 0$
- Establecer la solución inicial $x^0 = 0$.

- Establecer el residuo inicial $r^0 = y - Ax^0 = y$.
- Establecer el soporte inicial $S^0 = suppx^0 = 0$.
- Repetir
  - Establecer $k = k + 1$.
  - Escoger un i_0 tal que $min_c \parallel ca_{i_0} - r^{k-1} \parallel_2 \leq min_c \parallel ca_i - r^{k-1} \parallel_2$ para todo i.
  - Establecer $S^k = S^{k-1} \cup \{i_0\}$.
  - Calcular $x^k = argmin_x \parallel Ax - y \parallel_2$ sujeto a $suppx = S^k$.
  - Calcular $r^k = y - Ax^k$.
- Hasta $\parallel r^k \parallel_2 < \epsilon$.

Salida

- Solución aproximada x^k

Anexo D. Índice de vídeos utilizados en el proyecto

En este anexo se listan los vídeos de la base de datos usados en la realización de este proyecto, así como una breve descripción y la fuente donde poder consultar una versión en baja resolución de los mismos. Es importante recordar que los experimentos realizados en el proyecto no pueden reproducirse con esta versión de los vídeos, no obstante, la base de datos original esta disponible bajo petición. Los vídeos se encuentran disponibles en un DVD suplementario junto con esta memoria, así como en un fichero on-line.
Enlace: `https://drive.google.com/folderview?id=0Bxsb8iSM9CHNZ1ZQeml5TDFIWWs&usp=sharing`

D.1. Vídeos de entrenamiento

Para el entrenamiento de diccionarios se han utilizado diez vídeos con imágenes de diferente naturaleza capturadas. Una muestra de cada uno de estos vídeos puede verse en la Figura D.1. Listado de los vídeos:

- Aspas: aspas de un ventilador manual girando. En el centro, el dibujo de un balón de fútbol va girando a la misma velocidad.
  Resolución espacial: 512×512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Blink: parpadeo de un ojo humano.
  Resolución espacial: 512×512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Book: páginas de un libro de texto siendo pasadas.
  Resolución espacial: 512×512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Brain: objeto de goma con la forma de un cerebro rebotando en una mesa contra una cadena de bici.
  Resolución espacial: 512 × 512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Coin: moneda de 50 céntimos de Euro girando sobre una superficie.
  Resolución espacial: 512 × 512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Dado: dado de veinte caras girando sobre una superficie.
  Resolución espacial: 512 × 512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Dados: conjunto de dados de diferentes texturas cayendo y rebotando sobre una superficie.
  Resolución espacial: 512 × 512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Drop: flujo de agua, incluyendo gotas, cayendo de un cuentagotas hacia una superficie.
  Resolución espacial: 512 × 512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Globo: globo explotando bajo la influencia de una llama.
  Resolución espacial: 512 × 512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

- Rueda: Rueda de un coche de juguete girando.
  Resolución espacial: 512 × 512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

D.2. Vídeos originales del análisis

Para el análisis se han reconstruido vídeos diferentes a los vídeos de entrenamiento, también de diferente naturaleza entre ellos. Una muestra de cada uno de estos vídeos puede verse en la

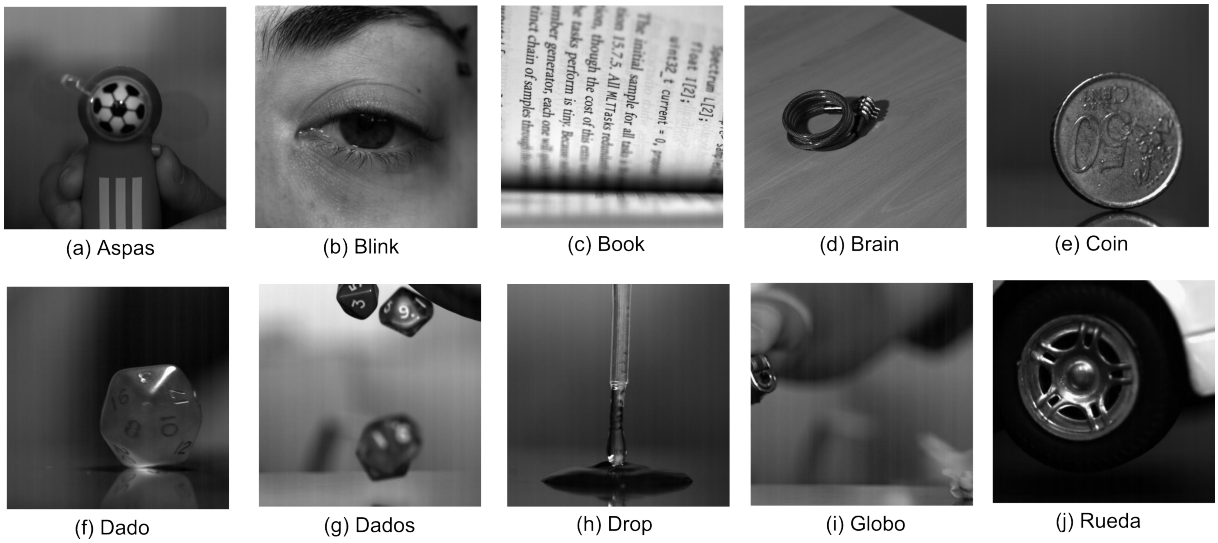

Figura D.1: Muestra de un fotograma representativo para cada uno de los vídeos utilizados en el entrenamiento de diccionarios.

Figura D.2.
Listado de los vídeos:

- City: filmación de una ciudad desde un helicóptero.
  Resolución espacial: 200 × 200 píxeles.
  Resolución temporal: 30 fps.
  Duración: 20 fotogramas.

- Coke: cocacola siendo vertida en un vaso desde una botella.
  Resolución espacial: 200 × 200 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 20 fotogramas.

- Foreman: persona hablando mientras mueve la cabeza.
  Resolución espacial: 200 × 200 píxeles.
  Resolución temporal: 30 fps.
  Duración: 20 fotogramas.

- LlamaHold: llama estabilizada tras ser prendida en un mechero.
  Resolución espacial: 200 × 200 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 20 fotogramas.

- LlamaStart: llama prendiendo en un mechero.
  Resolución espacial: 200 × 200 píxeles.

Resolución temporal: 1000 fps. Duración: 20 fotogramas.

- Spring: muelle rebotando en una superficie.
  Resolución espacial: 200×200 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 20 fotogramas.

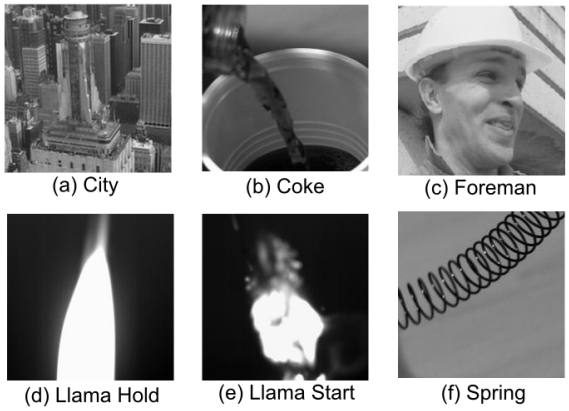

Figura D.2: Muestra de un fotograma representativo para cada uno de los vídeos utilizados en el análisis.

D.3. Vídeos reconstruidos

Como vídeos reconstruidos se presentan los vídeos resultado de la reconstrucción de los originales de la Sección D.2 con el diccionario por defecto. También se incluyen dos vídeos reconstruidos a mayor resolución del capítulo 7. Recordar que se pierden los píxeles de los bordes debido a la división con solapamiento.

Vídeos reconstruidos la sección de análisis
Listado de los vídeos:

- CityR
  Resolución espacial: 188×188 píxeles.
  Resolución temporal: 30 fps.
  Duración: 20 fotogramas.
- CokeR
  Resolución espacial: 188×188 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 20 fotogramas.
- ForemanR
  Resolución espacial: 188×188 píxeles.

Resolución temporal: 30 fps.
Duración: 20 fotogramas.

- LlamaHoldR
  Resolución espacial: 188 × 188 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 20 fotogramas.
- LlamaStartR
  Resolución espacial: 188 × 188 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 20 fotogramas.
- SpringR
  Resolución espacial: 188 × 188 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 20 fotogramas.

Vídeos reconstruidos en la sección de resultados
Listado de los vídeos:

- CokeFull
  Resolución espacial: 512 × 640 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 80 fotogramas.
- CokeFullR
  Resolución espacial: 500 × 628 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 80 fotogramas.
- LlamaFull
  Resolución espacial: 512 × 512 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.
- LlamaFullR
  Resolución espacial: 500 × 500 píxeles.
  Resolución temporal: 1000 fps.
  Duración: 200 fotogramas.

Anexo E. Caracterización de los vídeos utilizados durante el análisis

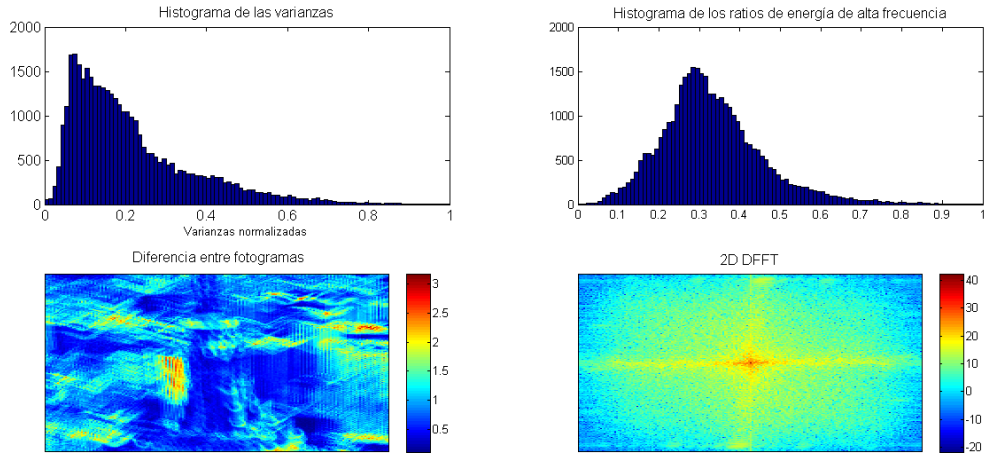

Figura E.1: Métodos de caracterización para el vídeo City. En la fila de arriba se pueden ver el histograma de las varianzas de cada bloque (izquierda) y el histograma del ratio de energía de alta frecuencia temporal (derecha). En la fila de abajo se puede ver la imagen calculada como la suma de diferencias entre fotogramas consecutivos (izquierda), y la transformada de Fourier bi-dimensional (derecha).

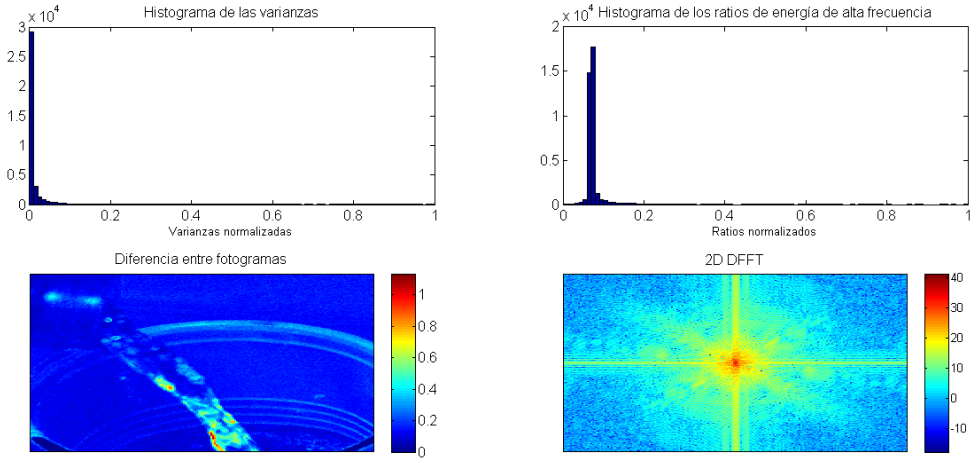

Figura E.2: Métodos de caracterización para el vídeo Coke. En la fila de arriba se pueden ver el histograma de las varianzas de cada bloque (izquierda) y el histograma del ratio de energía de alta frecuencia temporal (derecha). En la fila de abajo se puede ver la imagen calculada como la suma de diferencias entre fotogramas consecutivos (izquierda), y la transformada de Fourier bi-dimensional (derecha).

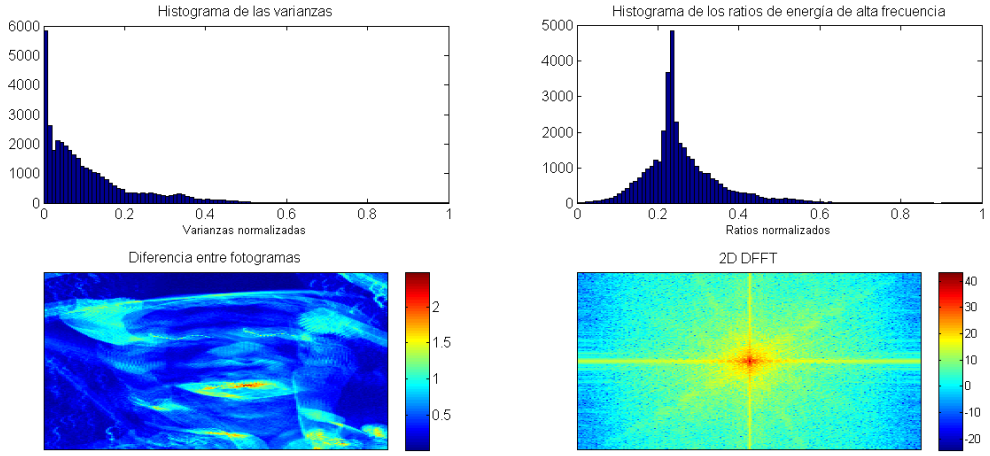

Figura E.3: Métodos de caracterización para el vídeo Foreman. En la fila de arriba se pueden ver el histograma de las varianzas de cada bloque (izquierda) y el histograma del ratio de energía de alta frecuencia temporal (derecha). En la fila de abajo se puede ver la imagen calculada como la suma de diferencias entre fotogramas consecutivos (izquierda), y la transformada de Fourier bi-dimensional (derecha).

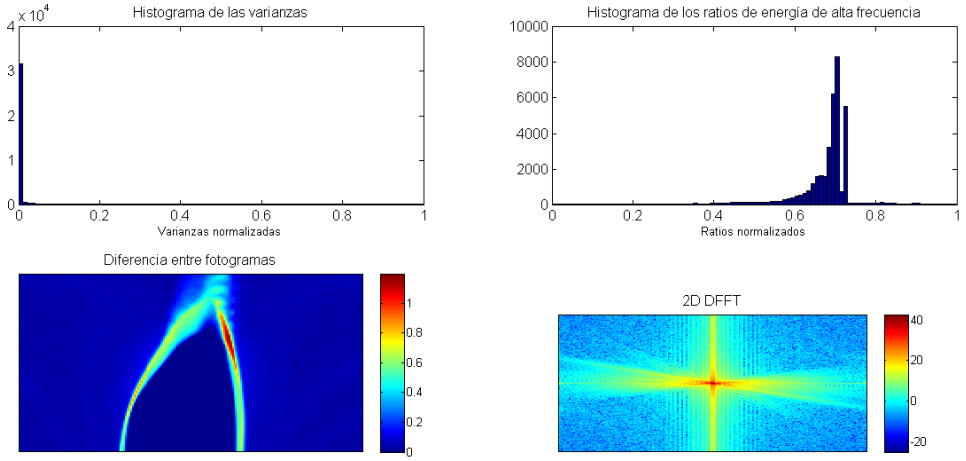

Figura E.4: Métodos de caracterización para el vídeo LlamaHold. En la fila de arriba se pueden ver el histograma de las varianzas de cada bloque (izquierda) y el histograma del ratio de energía de alta frecuencia temporal (derecha). En la fila de abajo se puede ver la imagen calculada como la suma de diferencias entre fotogramas consecutivos (izquierda), y la transformada de Fourier bi-dimensional (derecha).

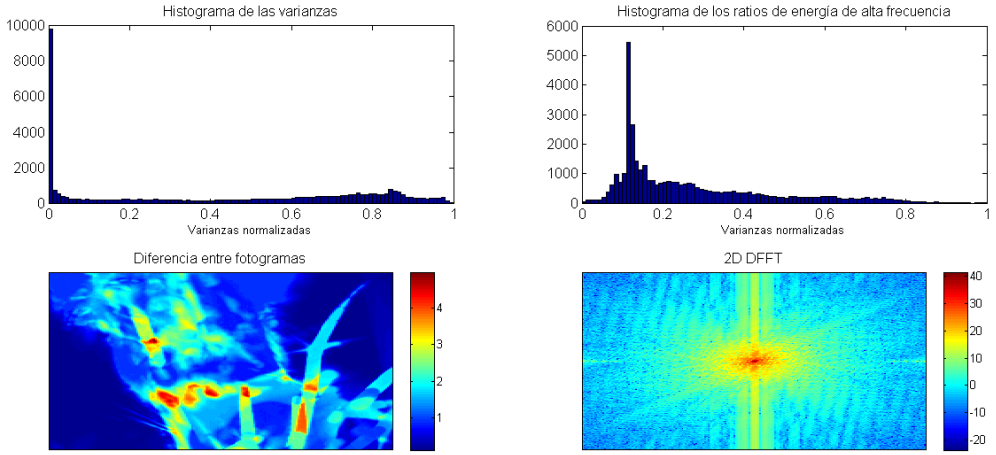

Figura E.5: Métodos de caracterización para el vídeo LlamaStart. En la fila de arriba se pueden ver el histograma de las varianzas de cada bloque (izquierda) y el histograma del ratio de energía de alta frecuencia temporal (derecha). En la fila de abajo se puede ver la imagen calculada como la suma de diferencias entre fotogramas consecutivos (izquierda), y la transformada de Fourier bi-dimensional (derecha).

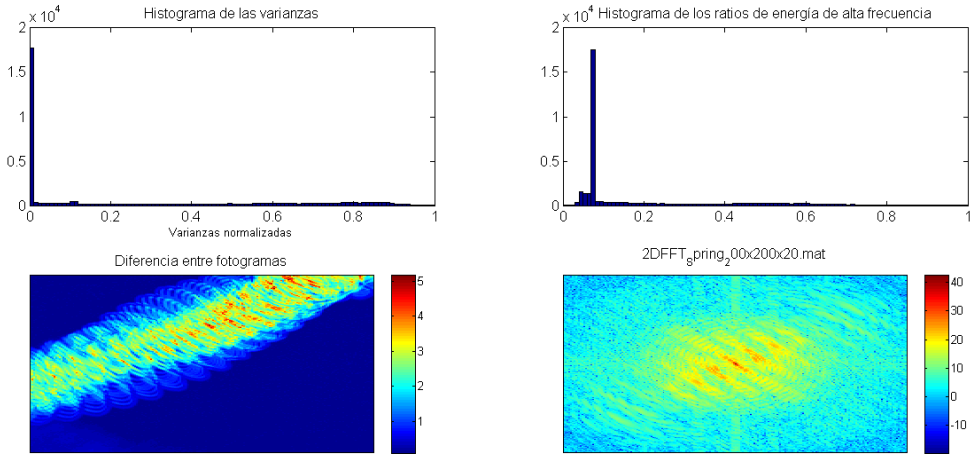

Figura E.6: Métodos de caracterización para el vídeo Spring. En la fila de arriba se pueden ver el histograma de las varianzas de cada bloque (izquierda) y el histograma del ratio de energía de alta frecuencia temporal (derecha). En la fila de abajo se puede ver la imagen calculada como la suma de diferencias entre fotogramas consecutivos (izquierda), y la transformada de Fourier bi-dimensional (derecha).