Identificación de la fuente de adquisición de imágenes/vídeos en escenarios abiertos usando técnicas de IA

IMPLANTACIÓN CORPORATIVA DE TECNOLOGÍAS, SERVICIOS Y SISTEMAS INFORMÁTICOS

Máster en Ingeniería Informática

Ignacio Gago Padreny

Facultad de Informática
Universidad Complutense de Madrid

Madrid, Junio de 2016

Índice General

Capítulo 1

Vídeos Digitales

En este capítulo se describen los principales conocimientos sobre vídeos relacionados con el objetivo principal de este trabajo. En primer lugar se detalla el proceso de generación de un vídeo y su composición basada en imágenes para luego hablar de los métodos más habituales de compresión para el almacenamiento del mismo. Una vez explicado este proceso se comentará cómo interviene el tipo de sensor en la extracción del ruido del dispositivo.

1.1. Proceso de generación de un vídeo digital

El proceso de generación de un vídeo digital está basado en transformar señales analógicas (funciones con dominio continuo y que toman valores en un conjunto continuo) en señales digitales, capaces de ser procesadas por un ordenador. Para convertir una señal analógica en una señal digital (conversión A/D) se utilizan dos técnicas: muestreo y cuantificación.

1.1.1. Muestreo

El proceso de muestreo o *sampling* consiste en transformar una señal con dominio continuo en otra de dominio discreto, de forma que se retenga el máximo posible de la información original de la señal analógica. Gráficamente se puede ver un ejemplo de muestreo en 3.1.

La técnica del muestreo ha sido ampliamente estudiada pues es usada en una gran variedad de campos y existen métodos y fórmulas matemáticas para determinar cotas inferiores que no elimenen información del original, sin embargo en este contexto hay cotas menos exigentes debido a la capacidad que tiene el cerebro para procesar visualmente un objeto. En el contexto de este trabajo hay dos tipos dife-

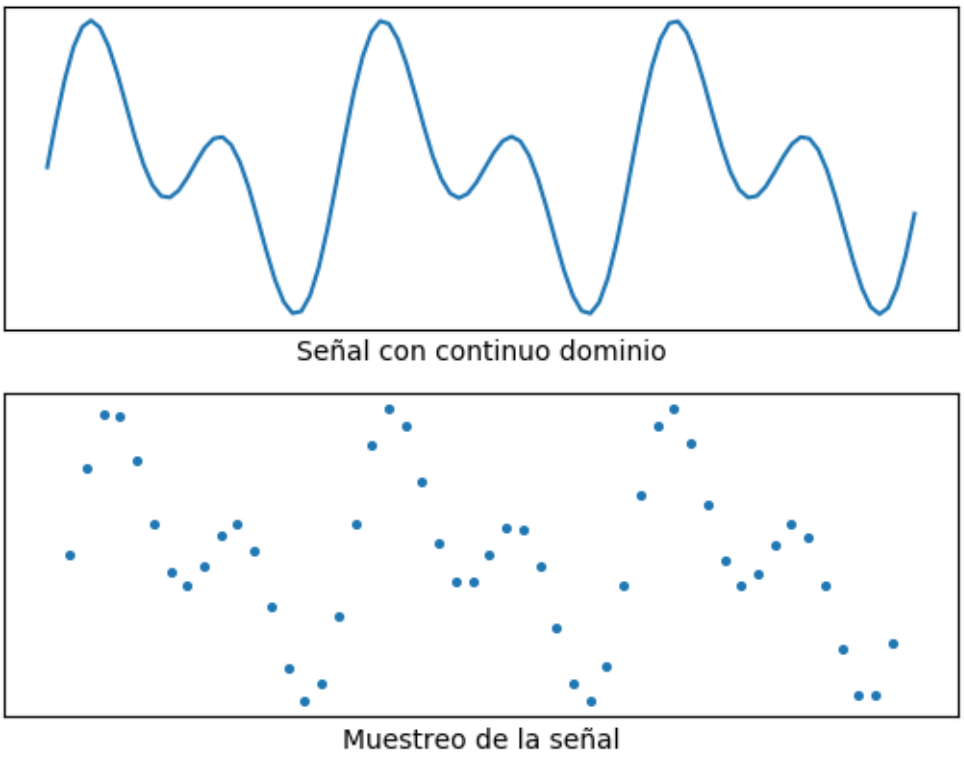

Figura 1.1: Muestro de una señal de dominio continuo

rentes que se deben realizar: uno asociado a las variables espaciales y otro asociado al tiempo. Ambos casos se basan en tener muestras muy cercanas de forma que la composición parezca continua y no discreta. El muestro de la variable temporal está relacionado con el número de imágenes por segundo que es capaz de procesar el ojo humano, estando entre 25 y 30 lo que el ojo ya percibe como continuo. Al discretizar la senãl analógica obtenemos un conjunto finito que podemos numerar y expresar en forma de una matriz de dos dimensiones, siendo cada una de las celdas un pixel (del inglés *picture element*). Para el número de filas y columnas elegido por el muestro se toma un múltiplo de dos, puesto que tiene por una parte la ventaja de favorecer el direccionamiento de las muestras y por otra de ser más eficientes para ciertos algoritmos como puede ser la transformada de Fourier. En el muestro también interviene la frecuencia de la señal original: una con baja frecuencia puede ser bien representada con una tasa de muestreo determinada, pero la misma puede ser no válida para una frecuencia alta, produciéndose el efecto que conocemos como solapamiento o *aliasing*. El teorema de Nyquist establece que utilizando una tasa

de muestro mayor al doble de la frecuencia original, se evita el *aliasing* y se puede recuperar la señal original a partir de la transformada. En 1.2 se puede observar como cuando la frecuencia de muestreo es suficientemente grande comparado con la frecuencia original (figura de arriba) se puede reconstruir la onda original, mientras que en la figura de abajo se observa que cuando no se cumple el Teorema de Nyquist se produce una pérdida de información que impide reconstruir la señal original[9].

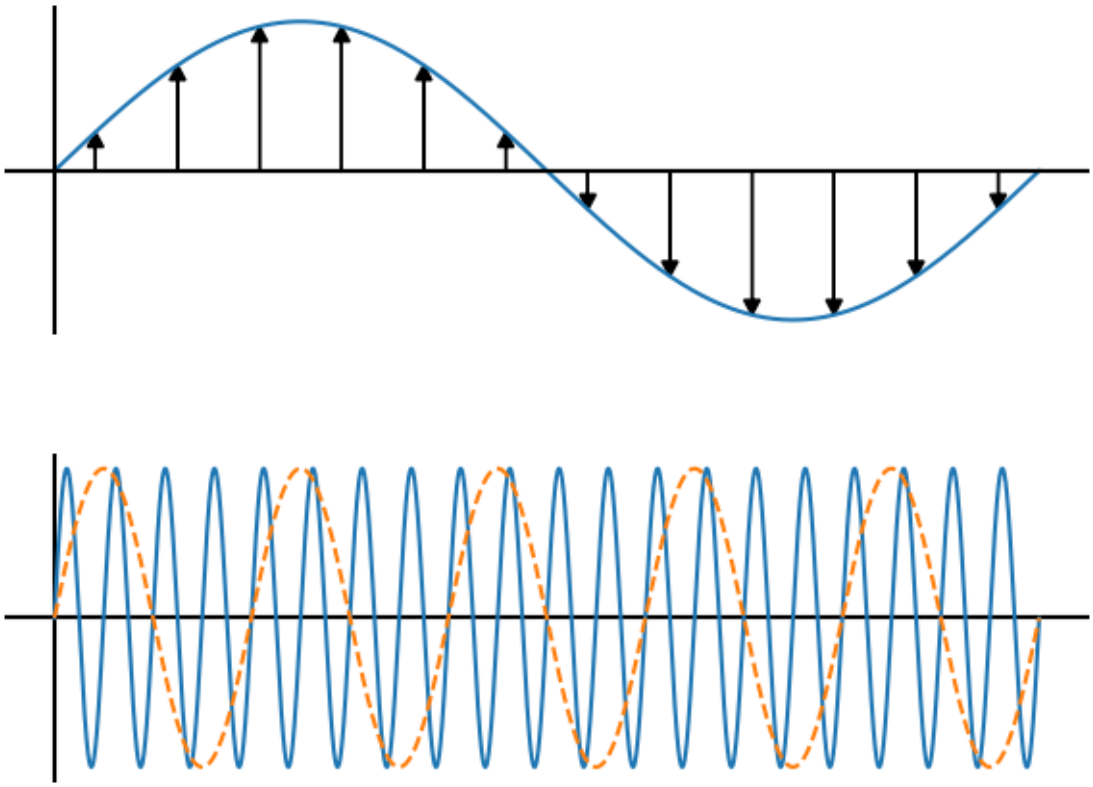

Figura 1.2: Frecuencia en el muestreo

1.1.2. Cuantificación

La cuantificación consiste en transformar el rango continuo de la señal analógica en un rango discreto. La intensidad que es una señal continua es transformada a un conjunto finito que son los valores que pueden tomar los pixels. De esta forma, mientras que con el muestro reducimos una variable espacial continua en una matrix, la cuantificación permite que la intensidad que capta la lente del dispostivo se pueda representar por un conjunto discreto de valores. De la misma forma que

en el muestreo, se suele utilizar un conjunto de cardinalidad potencia de dos. Para imágenes en color lo usual es trabajar con tres componentes cada uno de ocho bits, mientras que en las imágenes en blanco y negro se trabaja con un componente de ocho bits. Cabe destacar que este proceso no es reversible y está asociado a funciones no lineares, al contrario que el muestro, en el que partiendo de premisas no muy exigentes se puede reconstruir la señal analógica.

El proceso de cuantificación en vídeo se basa en aplicar frame a frame el método que se aplica en JPEG. En primer lugar se divide la imagen o frame en bloques disjuntos de 8x8 pixels, para cada uno de estos bloques B se calcula la transformada del coseno discreta (DCT) siguiendo[17]:

$$D_{ij} = \sum_{k,l=0}^{7} a_{kl}(i,j) B_{kl}$$

donde

$$a_{kl}(i,j) = \frac{1}{4} w(k) w(l) \cos \frac{\pi}{16} k(2i+1) \cos \frac{\pi}{16} k(2j+1) \tag{1.1}$$

y

$$w(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } k = 0 \\ 1 & \text{en caso contrario} \end{cases}$$

Aplicando DCT se transforma la fuente original en el dominio de las frecuencias. Los coeficientes a_{kl} de la ecuación 1.1, multiplicadores de los valores del bloque de la imagen, cumplen que a medida que nos distanciamos de la primera fila se incrementa la varianza, como podemos ver en 1.3. Además, a medida que nos distanciamos de la primera columna también crece la varianza. Por otra parte, los coeficientes DCT que se corresponden con frecuencias bajas son grandes en magnitud.

La matriz D con los coeficientes DCT es discretizada posteriormente utilizando una matriz de cuantificación Q, producto de una tabla de valores y cuantificación y una escala de cuantificicación, fija en el caso de una tasa de bits variable (VBR) y variable en el caso de una tasa de bits constante (CBR).

$$D_{ij} = round\left(\frac{D_{ij}}{Q_{ij}}\right), i, j \in \{0, \ldots, 7\}$$

En la matriz de cuantificación, cada elemento define el umbral bajo el cual un detalle en la imagen debe ser capturado como tal o descartado. De esta forma,

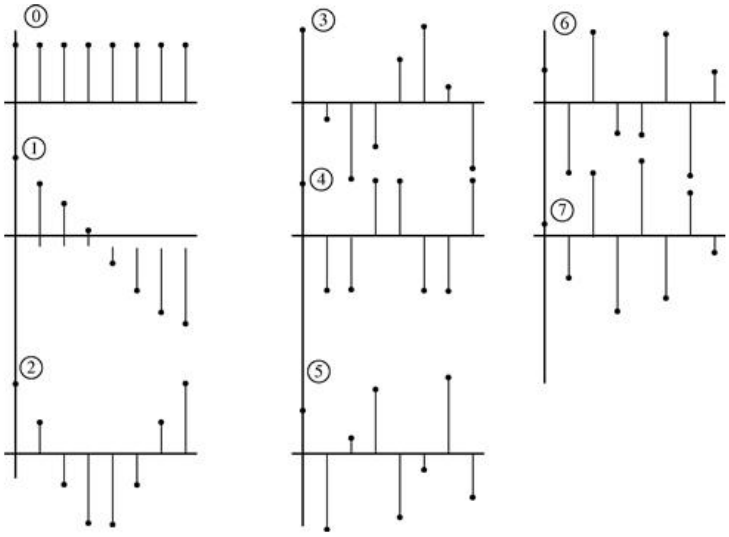

Figura 1.3: Varianza según la fila del bloque en la transformada DCT, [21]

a medida que nos alejamos del origen, ya sea horizontal o verticalmente, se exige un mayor coeficiente DCT para que el detalle sea relevante, puesto que se corresponden con valores de mayor frecuencia. Hay una gran variedad de matrices de cuantificación, calculadas normalmente en base a experimentos psico-visuales para determinar los umbrales DCT. Una matriz de cuantificación ampliamente utilizada es la siguiente[9]:

$$\begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix}$$

1.2. Almacenamiento de vídeos digitales

Del proceso descrito en la sección anterior, es fácil deducir que la cantidad de datos en señales visuales es grande. Una imagen en blanco y negro de dimensiones MxN con B bits para el nivel de resolución del gris supone un tamaño de NMB bits. Esto supone que una sola imagen de color de 512 x 512 x 8 ocupa cerca de 1MB. Esto implica que un vídeo con estas características y una tasa de muestro de 30 frames por segundo (el mínimo para que el ojo humano lo detecte como continuo)

requiere 23.6MB por segundo[8].

Esta gran cantidad de datos que sería necesaria para almacenar un vídeo no solamente supone un problema en cuanto a requisitos de memoria, si no también para el procesamiento y trasmisión de los mismos. Como consecuencia, es necesario reducir la cantidad de datos mediante algoritmos de compresión, que en el caso de vídeos están definidos por el comité MPEG (del inglés *Moving Pictures Expert Group*) de forma estándar e internacional. Como ya se ha comentado anteriormente, se puede ver un vídeo como una sucesión de imágenes o *frames*. Además de aprovechar la compresión de imágenes, en el caso del vídeo se tiene una redundancia temporal ya que el siguiente frame tiene mucho en común con el actual y los anteriores, factor que se aprovechará para reducir el tamaño.

Los algoritmos de codificación más utilizados han sido definidos por un grupo de expertos conocido como MPEG (del inglés *Motion Picture Expert Group*) y se basan, al igual que la mayoría de algoritmos de compresión de vídeo en el concepto llamado grupo de imágenes o GOP (del inglés *Group Of Pictures*). Un GOP de tamaño N está compuesto de N imágenes que pueden ser de tipo:

- Los **I-frames** o *intra-coded frames* se codifican de forma independiente, sin referencias a otros frames. Esto permite acceso aleatorio a los datos del vídeo, puesto que pueden ser decodificados sin necesitar de otros frames. Además de esto, tienen la ventaja de evitar la propagación de errores que se acarrea en la compresión de frames consecutivos al contener la mayor información de la escena por si solos, a costa de ocupar más que los otros tipos de frames. En cada GOP debe constar al menos un I-frame.

- Los **P-frames** son frames pronosticados, comprimidos basados en la diferencia que existe respecto de un I-frame o P-frame anterior.

- Los **B-frames** son frames bidireccionales que usan los datos de imágenes previas y posteriores de I-frames o P-frames.

- Los **D-frames** son frames de baja resolución que raramente se utilizan y que se obtienen decodificando el coeficiente *dc* de la transformada de coseno discreta (DCT) de los coeficientes de cada macrobloque.

En cuanto a la compresión:

- Los I-frames se comprimen mediante el uso de la transformada del coseno discreta y la cuantificación, de la misma forma que en el caso de imágenes,

puesto estos frames deben contener toda la información relevante de manera aislada. Se comprime por separado la luminosidad y la crominancia.

- En el caso de los P-frames la compresión depende de la similitud entre el frame en cuestión y los del grupo en que se encuentra. Si no se encuentra un frame adecuado, este deberá comprimirse del mismo modo que si se tratase de un I-frame. En caso de encontrarse un buen candidato, se computa el residuo entre ambos y se cuantifica utilizando la siguiente matriz:

$$\begin{bmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{bmatrix}$$

- Los B-frames utilizan el mismo procedimiento que los P-frames con la diferencia que también buscan similitud con frames posteriores en el grupo, y también pueden utilizar relación entre un frame anterior y uno posterior simultáneamente.

Una vez se tiene la cuentificación del frame, independientemente del tipo que sea, éste se almacena siguiendo una traza en forma de zig-zag1.4 y no de forma secuencial, agrupando los ceros correspondientes a los coeficientes de alta frecuencia en un mismo grupo. Posteriormente la codificación se realiza mediante el algoritmo de Huffman[23].

El procesamiento de un GOP no es secuencial, al existir relaciones bidireccionales entre cierto tipo de frames. Al empezar un GOP, en primer lugar se procesa el I-frame. El siguiente frame a procesar será de tipo P, puesto que solamente necesita de este I-frame. Una vez procesados estos dos frames, los B-frames que están en medio serán decodificados. El proceso sigue alternando el procesamiento de P-frames con B-frames intermedios, hasta finalizar el GOP en cuestión, como se puede ver en 1.5.

1.3. Procesamiento en sensores de imágenes

Existen principalmente dos tipos de sensores que se usan para capturar imágenes o vídeo: sensores CCD (del inglés *Charge Coupled Device*) y sensores CMOS (del

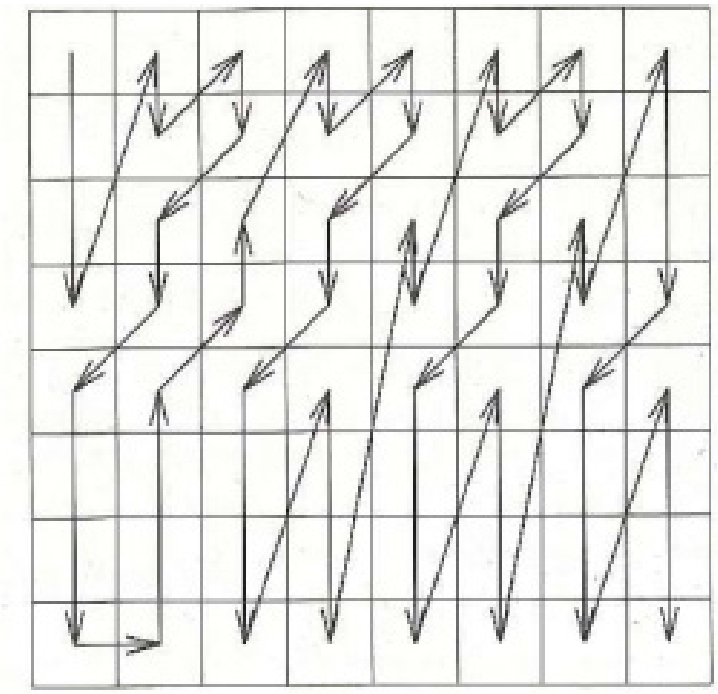

Figura 1.4: Método del zig-zag[7]

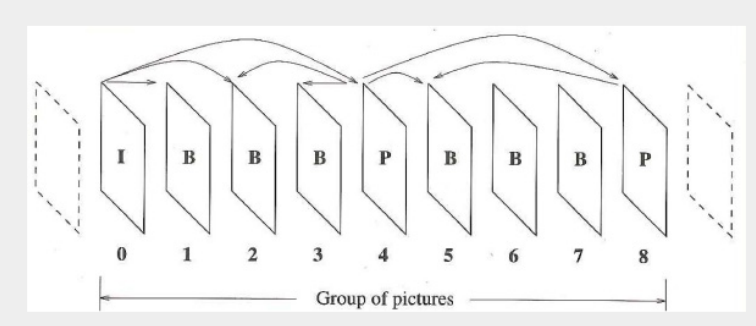

Figura 1.5: Procesamiento en GOP

inglés *Complementary Metal Oxide Semiconductor*). Ambos sensores se basan en el mismo principio que es capturar la máxima cantidad de luz que incide en el sensor y convertirla en una seãl eléctrica que será transformada posteriormente en digital. Los sensores CMOS tratan los píxeles de forma individaul mientras que los sensores CCD se basan en la propagación de carga eléctrica mediante condensadores. En la actualidad, los sensores CMOS son ampliamente utilizados, sobre todo en dispositivos móviles, ya que los sensores CCD necesitan un chip adicional y son más costosos y grandes que los CMOS, por lo que en lo que sigue se detallará el funcionamiento de los sensores CMOS[22].

Un sensor CMOS consiste en una matrix de sensores de pixels, cada uno de ellos compuesto de un fotodetector y un amplificador activo. Cada uno de estos sensores de pixels captura información de un píxel en uno de los tres colores primarios (rojo, verde y azul) puesto que se aplica un filtro de color conocido como CFA (*Color Filter Array*). Bayer es el CFA más utilizado, compuesto por un patrón de filtro que es la mitad verde, un cuarto azul y un cuarto rojo, debido a que el ojo humano es más sensible al color verde1.6[9].

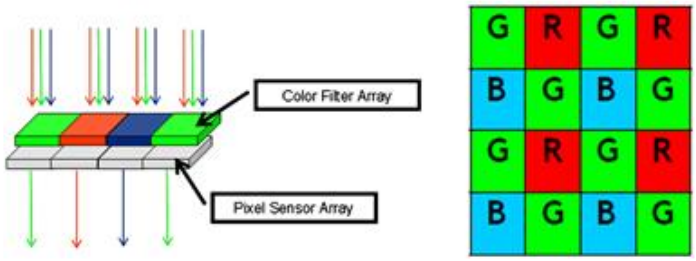

Figura 1.6: Bayern CFA, [9]

Tras la aplicación del filtro CFA para cada píxel se tiene únicamente información sobre un color, lo que implica que se tiene que llevar a cabo un proceso para estimar los valores de los otros dos componentes del color. Esta estimación se puede realizar mediante distintas técnicas, todas ellas basadas en utilizar los valores de los píxels cercanos. Se pueden usar métodos sencillos como el del vecino más próximo (el píxel toma el valor del píxel que le precede) o el bilinear (toma como valor la media de sus vecinos en vertical y horizontal más próximos) u otros más complejos como pueden ser splines cúbicas, método de mínimos cuadrados o filtros direccionales[8].

Tras la conversión Bayer-RGB hay varias funciones que en función del dispositivo y sensor pueden aplicarse como pueden ser corrección del color o corrección gamma.

Hay que tener en cuenta que en este proceso el hardware tiene una influencia considerable. Cada sensor a pesar de pertenecer al mismo fabricante tiene pequeñas imperfecciones o diferencias del resto que impactan directamente en la imagen obtenida. Esto es conocido como el ruido del sensor, que se aborda en la siguiente sección.

1.4. Extracción del ruido en imágenes

Los principales componentes del ruido en imágenes por imperfecciones del sensor son el FPN (*Fixed Pattern Noise*) y el PRNU (*Photo Response Non Uniformity*).

El ruido FPN se genera por la corriente oscura y depende también de la exposición y de la temperatura. Es un ruido independiente de las imperfecciones del sensor y es un ruido aditivo que es eliminado en algunas cámaras restando una capa de color negro.

El ruido PRNU es el mayoritario y es un ruido multiplicativo, lo que complica su eliminación. Está compuesto por dos ruidos: PNU (*Pixel Non-Uniformity*) y por defectos de baja frecuencia como pueden ser la configuración del zoom, y la refracción de la luz en las lentes. Es el primero de estos dos componentes, el PNU, el que tiene que ver con la fabricación de los wafers de silicio y las imperfecciones en el proceso de fabricación, lo que hace que sea un atributo único de cada sensor.

La extracción del PRNU se basa en aplicar una función a la imagen original que elimine el ruido de la imagen, obteniendo la imagen limpia de ruido. Al substraer la imagen original de la imagen sin ruido obtenemos el ruido. En [3] usan el algoritmo BM3D y en [4] proponen usar el algoritmo FSTV, en [1] usan transformadas de ondículas o *wavelets* para eliminar el ruido. Este ruido puede estar contaminado por agentes externos, en consecuencia se han desarrollado técnicas como *zero-mean*[6] o el filtro de Wiener[10].

Capítulo 2

Análisis forense en vídeos digitales

En este capítulo se presentan las principales técnicas forenses aplicadas a vídeos. A pesar de ser un campo ampliamente investigado, la gran cantidad de vídeo que se produce a diario y el crecimiento de aplicaciones de edición de vídeo para usuarios no expertos ha crecido exponencialmente en los últimos años, lo que hace que muchos de los algoritmos desarrollados queden desactualizados frente a nuevos técnicas antiforenses. La gran digitilización de la sociedad en los últimas décadas ha influido notablemente en procesos judiciales, especialmente posteriormente a 1978 puesto que tras la legislación deFlorida se admitían pruebas digitales (e-mails, fotografías o vídeos digitales, audios, etc.) como evidencia. Para garantizar que pruebas digitales puedan considerarse como evidencia de sucesos reales tiene que garantizarse que no haya sido manipulada y debe poder establecerse la fuente de adquisión de los datos digitales en cuestión.

En las siguientes secciones se describen en detalle la detección de manipulación de vídeos, la detección de doble compresión (un caso concreto que permite la detección de manipulación de vídeos) y la identificación de la fuente.

2.1. Manipulación de vídeos

Actualmente existen multitud de herramientas de edición de vídeo a disposición de usuarios no expertos, que pueden manipular la imagen de múltiples formas como introduciendo, quitando, repitiendo o copiando frames, además de operaciones que no añaden objetos al vídeo como son la rotación, el escalado o la aplicación de filtros. En cualquiera de los dos casos, a pesar de que no haya añadido o eliminado ningún objeto en frames, afectará a la codificación del vídeo.

La aplicación de técnicas forenses para imágenes para la detección de manipulaciones en vídeos no es recomendable. La estructura de los vídeos y los distintos tipos de frames del GOP permiten técnicas basadas en la consistencia temporal incapaces de ser detectadas por medio de análisis estáticos de los frames considerados como imágenes, además de ser algoritmos computacionalmente costosos y de ser incapaces de detectar la inserción o eliminación de frames[12].

En [24] analizan el problema de la duplicación de frames y utilizan la correlación como medida de similitud. En primer lugar eligen un conjunto de segmentos candidatos de entre todos los del vídeo, reduciendo el espacio de búsqueda, para luego calcular la medida de similitud entre ellos a partir de histogramas de color y decidir si se ha producido duplicación o no. Para los experimentos han utilizado un conjunto de 15 vídeos manipulados y el algoritmo ha mostrado una precisión media del 85 %.

En [29] se centran en la detección de eliminación o duplicación de frames consecutivos. Para ello se basan en un término derivado de la velocidad de partículas en imágenes o PIV (del inglés *Particle Image Velocity*), cuya idea es comparar frames adyacentes y estimar el desplazamiento causados por la separación en el tiempo. La duplicación o eliminación de frames consecutivos contribuye a aumentar este desplazamiento. Para decidir si un desplazamiento concreto es debido a una manipulación, utilizan el test de Grubbs[30] que para una distribución normal permite detectar *outliers*. En los experimentos parten de 40 vídeos a partir de los que generan otros 40 eliminando frames y otros 40 duplicando frames. La precisión media es del $96,3\,\%$ con una tasa de falsos positivos del 10 %.

En [31] se basan en que la correlación entre los coeficientes de valores grises es consistente en vídeos pero cuando se produce una falsificación desaparece esta consistencia. Primero extraen la consistencia de la correlación de los coeficientes de los valores grises para posteriormente clasificar las características mediante SVM (del inglés *Support Vector Machine*). En los experimentos parten de una base de datos de vídeos originales y crean otras cuatro bases de datos a partir de los originales con las siguientes manipulaciones: insertando 25 frames, insertando 100 frames, eliminando 25 frames y eliminando 100 frames. Para entrenar la SVM utilizan 480 vídeos originales y 480 vídeos falsificados, dejando 118 originales y otros tantos manipulados para pruebas, consiguiendo precisiones por encima del 90 %.

Los mismos autores, en [32] utilizan otro método basado también en la consistencia temporal entre frames. En este caso en lugar de utilizar la correlación entre los coeficientes grises, utilizan la consistencia medida mediante el flujo óptico de Lucas-Kanade que permite determinar el movimiento de un objeto dentro de una secuencia de frames. Para los experimentos utilizan la mismas bases de datos que en el caso anterior y una SVM para la clasificación, consiguiendo una precisión media superior también al 90 %.

En [40] se basan en el flujo óptico pero lo calculan siguiendo Horn-Schunck en lugar de Lucas-Kanade. Extraen únicamente los frames tipo I y tipo P y extraen como características el PRG (del inglés *Prediction Residual Gradient*) y el OFG (del inglés *Optical Flow Gradient*), el primero centrado en variaciones en la posición de objetos mientras que el segundo está centrado en cambios en la luminosidad. Estas dos características son comparadas con unos umbrales determinados empíricamente para detectar picos, cuando los picos sean continuos se tratará de una manipulación. El método ha mostrado una precisión de un 86 % en las pruebas realizadas.

En [39] utilizan también el flujo óptico de Lucas-Kanade para detectar inconsistencias en el caso de manipulaciones de tipo inserción o eliminación de frames. En lugar de computar la correlación entre frames del flujo óptico primero utilizan un estadístico que resume la información de los vectores resultantes de Lucas-Kanade para comprobar la consistencia con estos. Cuando se detectan irregularidades en base a este estadístico calculan la correlación entre los vectores completos del flujo Lucas-Kanade. Utilizan para las pruebas un total de 115 vídeos con una precisión en torno al 90 %.

Algunos autores han desarrollado métodos basados en la extracción de algún tipo de huella a partir de los frames que componen el vídeo para detectar anomalías en frames con huellas significativamente distintas. Estos métodos tienen el incoveniente de que los vídeos comprimidos pierden mucha información sobre la huella, y solamente han demostrado dar buenos resultados en vídeos no comprimidos que suele ser poco usual. En [13] obtienen el PRNU de los primeros frames que componen el vídeo y utilizan distintas medidas para detectar ataques como inserción de *frames*, inserción de objetos y replicación de frames mediante la correlación entre el PRNU de referencia del vídeo y el ruido de un frame en concreto, la relación entre el ruido de dos frames consecutivos o la relación entre dos frames consecutivos. En [14] utilizan en lugar del PRNU un ruido que solamente es aplicable a los sensores CCD,

el ruido del fotón en disparo. Además, su método solamente es aplicable a vídeos grabados de forma estática sin la cámara en movimiento lo cual restrige mucho el ámbito de aplicabilidad del algoritmo. En el caso en el que se den las premisas de las que parten el método tiene una precisión del 97 %.

Aprovechando la consistencia entre frames de tipo temporal, algunas investigaciones se han centrado en aspectos de tipo geométrico como pueden ser que las propiedades físicas o de iluminación sean reales. En [15] se utilizan técnicas geométricas para detectar trayectorias imposibles de objetos en vuelo libre. Para ello, se modeliza el movimiento parabólico de estos objetos en tres dimensiones para proyectar este modelo posteriormente en dos dimensiones y compararlo con la trayectoria de ese mismo objeto en el vídeo. El método desarrollado es válido tanto para cámaras estáticas como para cámaras en movimiento y se han realizado experimentos con diversos vídeos ya sea generados por ellos u obtenidos de plataformas de compartición de contenido en los que han medido el error medio entre la trayectoria real y la trayectoria estimada mediante su procedimiento para ser capaces de clasificar cuando una trayectoria ha sido falseada, sin embargo no han datos sobre la precisión del algoritmo en cuestión. Hay que tener en cuenta que esta técnica solamente puede ser utilizada en vídeos en los que exista un objeto que describa una trayectoria parabólica y por tanto no es válida para cualquier vídeo.

Sin embargo, la mayoría de trabajos sobre la detección de manipulación de vídeos están basados en detectar la re-comprensión o doble compresión de un vídeo, puesto que al ser editado se vuelve a comprimir por segunda vez.

2.2. Doble compresión

Gran parte de los estudios sobre doble compresión se centran en vídeos con formato MPEG y utilizan las mismas ideas que en la detección de doble compresión de imágenes JPEG. En concreto, la re-cuantificación afecta de los coeficientes con un paso de cuantificación distinto del original afecta al histograma de los coeficientes DCT[12]. Estos coeficientes pueden ser aproximados como[16]

$$Y_{Q_1,Q_2} = \Delta_2 sign(Y) round\left(\frac{\Delta_1}{\Delta_2} round\left(\frac{|Y|}{\Delta_1}\right)\right)$$

siendo Δ_1 y Δ_2 el tamaño del paso en la primera y segunda compresión, respectivamente.

En [17] muestran como la relación que hay entre Δ_1 y Δ_2 influye en el histograma

creando un máximo característico. Intuitivamente, la idea es que al descomprimirse la imagen, modificarse una porción de la misma y volverse a comprimir, esa porción modificada mostrará trazas de una sola compresión mientras que el resto de la imagen tendrá rasgos de doble compresión.

En [25] presentan un método para el caso en el que se ha utilizado la misma matriz de cuantificación en ambas compresiones, basado en el número de coeficientes DCT distintos que hay en una compresión, doble compresión y triple compresión.

En [28] utilizan un conjunto de clasificadores binarios entrenados con distintas combinaciones entre Δ_2 que es conocida (se puede leer directamente de los datos del vídeo) y posibles Δ_1, para luego tomar el voto de la mayoría con las características concretas del vídeo en cuestión.

En [36] se basan en aplicar que se obtiene lo mismo si se comprime la imagen una vez con Δ_1 que si se comprime dos veces con la misma matriz de cuantificación Δ_1. De esta forma, dada la imagen original vuelven a comprimirla con distintas matrices hasta encontrar una que cumpla que en la mayoría de los bloques codificados se htenga la imagen original, obteniendo así la matriz de cuantificación que se utilizó en la última compresión. La manipulación se detecta en el caso de que existan algunos bloques distintos en la imagen comprimida que en la original puesto que entonces la última compresión no ha sido aplicada por igual en todos los bloques y ha existido una doble compresión. Para la experimentación contaron con un conjunto de 1338 imágenes y dentro de las pruebas que realizaron la peor precisión media que obtuvieron fue del $88,7\,\%$. Es importante tener en cuenta que si la imagen ha sido manipulada y se ha utilizado la misma matriz de cuantificación que en la primera compresión, este método no logrará detectarlo.

En [37] se extraen los coeficientes DCT para luego calcular las diferencias entre ellos en cuatro direcciones: horizontal, vertical, diagonal mayor y diagonal menor. Tras obtener estas cuatro matrices se truncan algunos elementos basados en ciertos umbrales para luego ser modeladas cada uno por medio de un proceso aleatorio de Markov de primer orden. Tras algunas transformaciones sobre estos procesos de Markov, se crea un array de características que será tratado por algoritmos de *machine learning* puesto que la doble compresión conlleva unos errores de redondeo que dejan muestras estadísticas caracterizables por Markov. Para la experimentación generan 5040 vídeos con la misma estructura GOP y con distintas combinaciones Δ_1 y Δ_2, con una precisión superior al $90\,\%$.

Otra estrategia ampliamente utilizada se basa en que mientras que los coeficientes DCT de una imagen siguen una distribución Laplace[34] o una distribución Cauchy[35], también siguen la ley de Benford[33], esto es, el primer dígito significativo es d con probabilidad $\log_{10}\left(1+\frac{1}{d}\right)$. Si los coeficientes DCT se alejan significativamente de esta distribución entonces podemos concluir que se ha utilizado doble compresión. Muchas investigaciones han utilizado procedimientos basados en la ley de Benford aplicados al caso de doble compresión JPEG en imágenes, extensibles al caso de vídeo.

En [19] parten de la hipótesis que el factor de multiplicación q_1 de la tabla de cuantificación y el factor q_2 de la segunda tabla de cuantificación varían mientras que la tabla se mantiene la misma. En el caso de que $q_1 > q_2$ se observan anomalías en el histograma de los coeficientes DCT de tipo AC (frecuencia distinta de cero en ambas dimensiones espaciales) distintos de cero y es posible detectar la doble compresión.

En [18] aplican la ley de Benford para un subconjunto de las frecuencias del DCT que identifican más sensible al número de compresiones y utilizan un multiclasificador compuesto de N clasificadores SVM $S_k(k = 1, \ldots, N)$ donde S_k es un clasificador binario que detecta si la imagen ha sido comprimida k veces o no. De esta forma el número de compresiones que ha sufrido la imagen se toma como el clasificador con mayor k que haya detectado que ha sido comprimido k veces. Para la experimentación se ha trabajado con un conjunto de prueba de 100 imágenes y un conjunto de test de 10 imágenes, donde han obtenido una precisión del 94 % considerando que como máximo podía haber $N = 4$ compresiones.

En [38] aplican la ley de Benford para vídeos puesto que la codificación MPEG para vídeos y JPEG para imágenes comparten el mismo proceso. Observan que cuando se utiliza VBR los coeficientes AC distintos de cero de los I-frames doblemente comprimidos se alejan de la ley de Benford solamente cuando la escala de cuantificación utilizada en Δ_2 es menor que la utilizada en Δ_1. En el caso de CBR sí que aprecian compartamientos anómalos sin distinción de casuística. Para formalizar lo anterior, utilizan clasificadores binarios SVM en los que tratan como unidad un GOP decidiendo que existe doble compresión si el porcentaje de frames detectados como doblemente comprimidos excede un umbral determinado, obteniendo una precisión media superior al 95 %.

Los métodos descritos arriba se basan en extender técnicas desarrolladas para imágenes para el caso de vídeo. Sin embargo, existen otros algoritmos que apro-

vechan la alteración de la estructura GOP de los vídeos. Dentro de un GOP, los P-frames están correlacionados con el I-frame inicial, de forma que en caso de existir doble compresión los frames que cambien de GOP, ver figura 2.1 mostrarán ciertas características estadísticas.

En [20] se centran en el error de movimiento o compensación de movimiento de los P-frames, esto es, la transformación que se debe aplicar a un frame de tipo I o de tipo P anterior para obtener el P-frame en cuestión. Al deshacerse la secuencia original GOP, existirán P-frames cuyo frame de referencia haya cambiado y el error de movimiento en ese caso es mayor. En este trabajo no se describe el método para determinar el umbral que determine si el error de movimiento corresponde con un cambio en la estructura GOP ni tampoco se dan resultados concretos sobre la precisión del algoritmo.

En [26] analizan las características periódicas del *string* de bits de datos y del *skip macroblocks* para todos los I-frames y P-frames. Los *skip mabroblocks* son usados en P-frames y B-frames y no contienen información, correspondientes a macrobloques en los que no se producen cambios respecto del I-frame sobre el que se codifican. Muestran como al cambiar la estructura GOP el número de *skip macroblocks* decrece puesto que el I-frame original en el que se basaba el P-frame era antes un P-frame.

En [27] utilizan el número de *inter-coded macroblocks* y de *skip macroblocks* de cada frame modelizados como $i(n)$ y $s(n)$, cuando se produce un pico en $s(n)$ hay una alta probabilidad de doble compresión y el I-frame del que toma los valores fuera anteriormente un frame de tipo P, utilizando un razonamiento análogo al caso anterior.

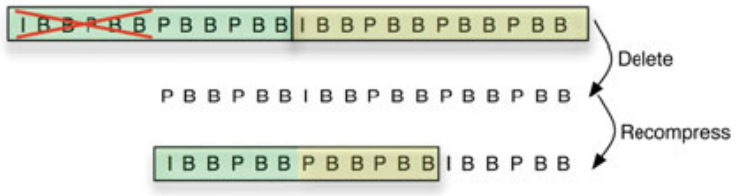

Figura 2.1: Reestructuración de GOP tras doble compresión, [12]

2.3. Identificación de la fuente

La identificación de la fuente de adquisición es de vital importancia para muchos procesos judiciales, podría compararse con las pruebas balísticas para identificar un arma. Es por esto por lo que la identificación de la fuente en imágenes ha sido am-

pliamente estudiado por académicos en los últimos años con buenos resultados. Esta sección se restringe a la identificación de la fuente entendido como la identificación del modelo fuente en dispositivos móviles y no engloba otras temáticas como podría ser distinguir entre gráficos generados por ordenador o capturados.

Existen muchas menos investigaciones sobre esta materia en vídeo que en lo referente a imagen, a pesar de que un vídeo se descompone como una secuencia de frames. Sin embargo, la menor resolución en vídeo frente a imagen y las altas compresiones que se utilizan hacen que se pierde mucha información sobre la huella.

En [42] extran una serie de fotogramas del vídeo en base a la luminosidad para extraer el PRNU mediante la descomposición *wavelet* de Daubechies de cuarto nivel a los que se aplica el filtro de Wiener. Se computa la correlación entre el ruido de cada frame para posteriormente evaluarlo mediante PCE (del inglés *Peak-to-Correlation Energy*). Se utiliza un método de clasificación en el que las imágenes a analizar son caracterizadas en uno u otro grupo según el PCE.

En [43] tratan el vídeo como una secuencia de N frames, para cada uno de esos frames extraen el PRNU y utilizan el estimador de máxima verosimilitud para identificar el PRNU del vídeo. Para decidir si dos vídeos fueron tomados por la misma cámara se basan en la covarianza normalizada y en el PCE: si provienen del mismo dispositivo entonces el PCE es grande por el pico en la covarianza normalizada y en caso de no provenir de la misma fuente la covarianza normalizada parecerá ruido blanco. En las pruebas se utilizaron 25 cámaras y muestra como el nivel de compresión del vídeo es crucial para el algoritmo, cuanta mayor compresión menor calidad y más tiempo de vídeo (en algunos casos 10 minutos de vídeo) se necesita para obtener un PRNU suficientemente bueno, lo que hace que este método no sea efectivo para vídeos de corta duración grabados por móvil.

En [44] utilizan un subconjunto de los coeficientes AC de la transformada DCT formado a partir de tres índices p, q y r que toman 8 orientaciones diferentes. Para cada una de esas orientaciones se calculan 9 estadísticos en base a la relación de orden entre p y q y entre r y q lo que da un total de 72 estadísticos diferentes que denominan caracerísticas CP, también utilizados en otros trabajos de estegoanálisis, que utilizarán como *input* para un clasificador de tipo SVM. Para las pruebas utilizan 4 modelos de cámara diferentes y 10 vídeos de cada una de ellas, obteniendo una precisión del 100 %.

En [41] utilizan características propias de la codificación MPEG-2, características relacionadas con la tasa de bits, los factores de cuantificación y los vectores de movimiento. Tanto la tasa de bits como los factores de cuantificación y los vectores de movimiento no son parámetros fijos en el estándar MPEG-2, cada fabricante establece unos en concreto según el sensor. Tras extraer estas características, se utiliza un clasificador SVM entrenado. Para las pruebas utilizan vídeos de ocho diferente codificadores y obtienen precsiones por encima del 86 %. Hay que tener en cuenta que vídeos obtenidos de cámaras que compartan el mismo codificador de MPEG-2 no serán clasificados como distintos por lo que este método solamente sirve para garantizar que dos vídeos provienen de distinta fuente.

En [5] se basan en que dentro de los canales RGB el verde es el que tiene más información sobre la huella. Por ello extran el canal verde de la imagen y mediante interpolación bilineal redimensionan los frames a tamaño 512x512 a los que extraen el ruido mediante *soft-thresholding*. El PRNU del vídeo lo obtienen como la media de los ruidos de cada frame y los clasifican utilizando la correlación como medida de similitud. En las pruebas muestra cómo los resultados de este proceso con el G-PRNU (*Green PRNU*) son mejores que con el PRNU.

Capítulo 3

Clustering

En este capítulo se describe el clustering y dos métodos principales. También se comentan varios métodos para elegir el número óptimo de clusters.

Clustering es una técnica de aprendizaje no supervisado que a partir de un conjunto de datos y una medida de similitud o distancia entre ellos los agrupa en distintas clases o clusters de forma que elementos que están en el mismo clúster son más parecidos entre ellos que con los de otro clúster distinto.

En primer lugar se describen los algoritmos de clustering más utilizados y posteriormente se comentarán métodos para elegir el número de clases o clusters óptimo.

3.1. K-means

El algoritmo de clustering K-means agrupa n elementos en k clusters S_k con el objetivo de minimizar la suma del error cuadrático intra-cluster:

$$\sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

donde μ_i es la media de las distancias entre los elementos en S_i, también conocidos como centroides.

La suma del error cuadrático intra-cluster o inercia es un indicador de la cohesión de los clusters. A mayor cohesión, menor distancia existirá entre los elementos de cada clúster y por tanto también del centroide. Este indicador sin embargo tiene ciertas desventajas:

- Asume que los clusters se modelan como esferas, al estar basado en k centroides y minimizar la distancia euclídea, lo que implica misma varianza entre clusters. Ver **??**.

- No es una métrica normalizada

- Muy sensible a outliers al utilizar la distancia al cuadrado

- No tiene en cuenta la densidad de cada cluster. Implícitamente asume que al ocupar cada clúster el mismo área cada cluster tiene que tener el mismo número de puntos. Ver 3.2.

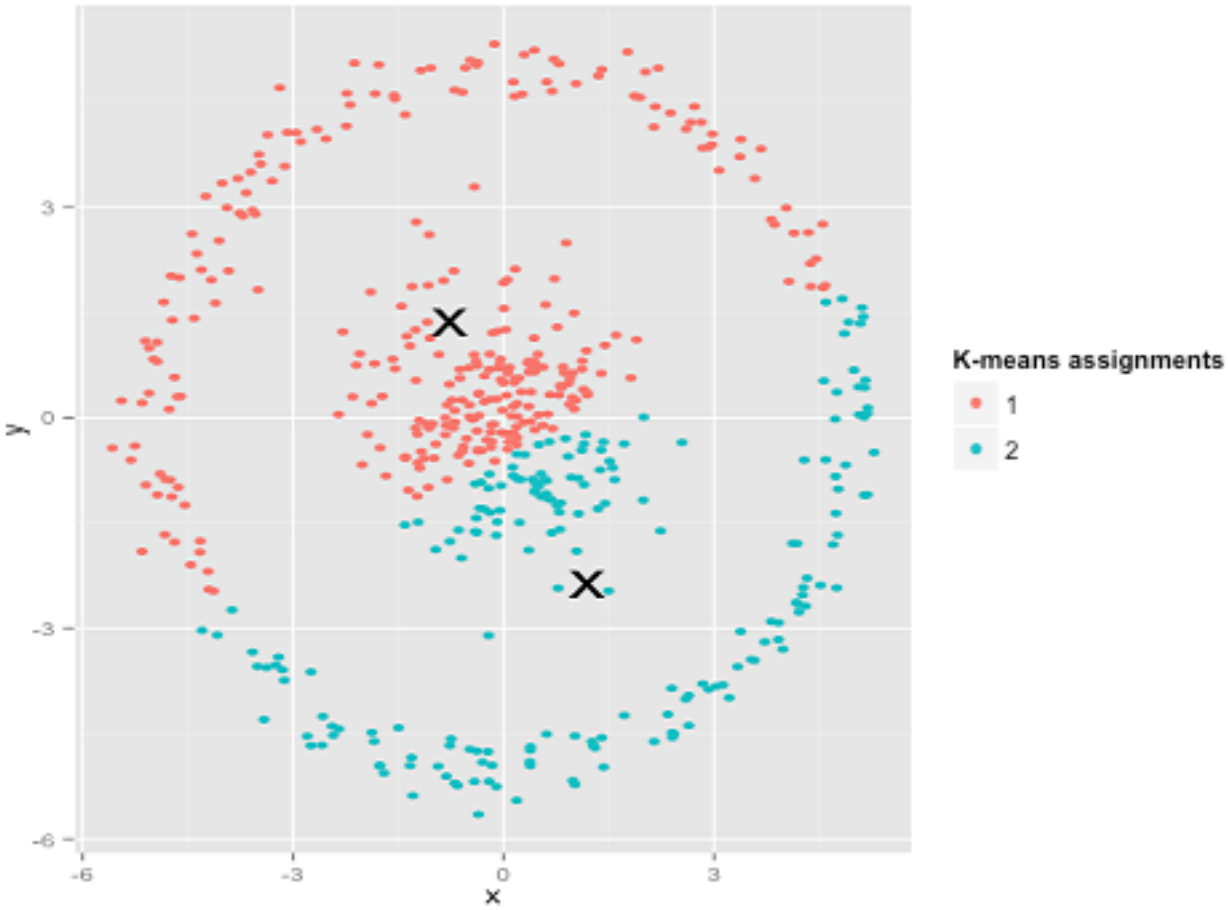

Figura 3.1: K-means frente a varianza en clusters

Dado un conjunto de n elementos $\{x_1, x_2, \ldots, x_n\}$ K-means empieza con una fase de inicialización en la que se escogen k centroides, $\{c_1, c_2, \ldots, c_k\}$. Una vez elegidos los centroides, itera de la siguiente forma:

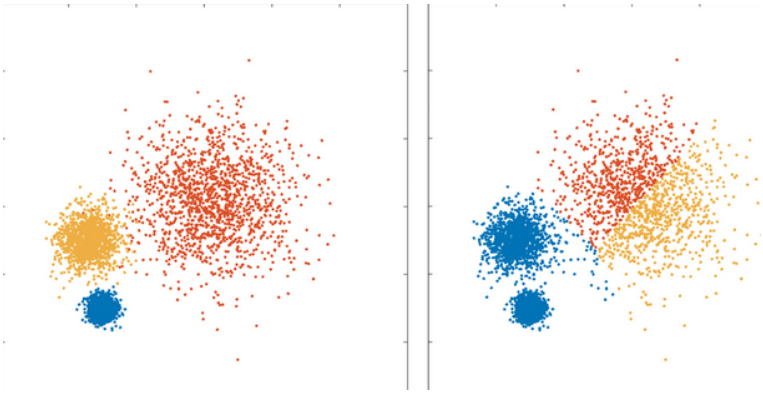

Figura 3.2: K-means no es sensible a la densidad de puntos

- **Etapa de asignación**: asigna cada elemento al centroide que minimiza la distancia euclídea al cuadrado, el cluster S_i que tiene como centroide c_i es

$$S_i = \{x_j : \|x_j - c_i\|^2 \leq \|x_j - c_p\|^2 \; \forall p, 1 \leq p \leq k\}$$

- **Actualización de los centroides**: serán la media de los elementos del cluster correspondiente.

$$c_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$$

Cuando en la etapa de asignación no se producen cambios entre dos iteraciones consecutivas, el algoritmo termina.

La inicialización de los centroides es un aspecto muy relevante en el algoritmo: K-means converge cuando encuentra un óptimo local por lo que se han desarrollado diversos métodos de inicialización de forma que se encuentre el óptimo global:

- Aleatorio: se asigna de forma aleatoria un elemento a cada cluster y posteriormente se calculan los centroides en base a esta asignación. Este método ubica los centroides cerca del centro del conjunto de datos.

- Forgy: se escogen al azar k elementos del conjunto y se utilizan como centroides. Este método tiende a dispersar los centroides iniciales.

- MacQueen: escoger al azar k elementos del conjunto y tratarlos como centroides. Asigna cada elemento al cluster con el centroide más próximo y recalcula los centroides, que serán los centroides de inicialización para el algoritmo.

- K-means++: propuesto en el año 2007. Funciona de la siguiente forma:

1. Se elige un centroide elegido aleatoriamente sobre el conjunto de observaciones.
2. Se calcula la distancia al cuadrado entre cada observación y el centroide seleccionado.
3. Se elige otro punto al azar como segundo centroide, la probabilidad que tiene cada observación de ser elegido es proporcional a la distancia al cuadrado calculada en (2).
4. Repetir (2) y (3) hasta tener k centroides.

En los últimos años se ha utilizado ampliamente la inicialización mediante K-means++, además de ejecutar varias veces el algoritmo con distintos centroides para evitar óptimos locales.

3.2. Clustering jerárquico

El clustering jerárquico se refiere a una familia de algoritmos de clustering que construyen clusters de forma anidada al fusionarlos (clustering aglomerativo) o dividirlos (clustering divisivos).

Cuando se trata de clustering aglomerativo, dado un conjunto de n elementos cada uno constituirá un cluster y en cada iteración se fusionarán dos clústers de forma que se minimize la medida de distancia o similitud especificada, terminando el algoritmo cuando haya 1 cluster.

En el caso de clustering divisivo, dado un conjunto de n observaciones se agrupan en un único cluster y se producen sucesivas divisiones, terminando cuando haya n clusters.

En cualquiera de los dos casos, las sucesivas iteraciones son representadas a través de un dendrograma. Como se puede ver en 3.3 un dendrograma es un diagrama en forma de árbol. En este caso muestra un algoritmo de clustering aglomerativo en el que los elementos $\{a, b, c, d, e, f\}$ se han ido agrupando en sucesivas iteraciones hasta terminar con 1 cluster.

En lo sucesivo se contemplará el clustering aglomerativo, los conceptos y métodos para el caso divisivo son análogos. Para decidir cómo agrupar los clusters de forma iterativa es necesario definir una medida de similitud entre clusters, de forma que clusters similares se agrupen antes que clusters distintos. En la primera

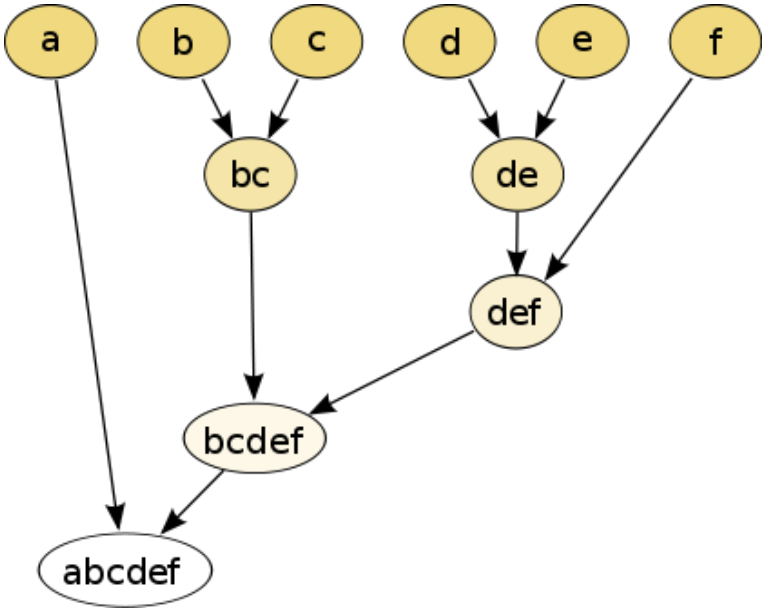

Figura 3.3: Dendrograma[46]

iteración en la que todos los clusters están compuestos por un único elemento se puede especificar una distancia, en el resto de iteraciones será necesario definir un criterio de enlace o *linkage criteria* que modele cómo medir la similitud entre dos clusters en los que al menos uno de ellos está compuesto por más de una observación.

Se puede especificar cualquier distancia d, mientras cumpla las propiedades de distancia desde un punto de vista matemático, que son:

- $d(x, y) \geq 0$ y $d(x, y) = 0 \iff x = y$
- $d(x, y) = d(y, x)$ o propiedad simétrica
- $d(x, z) \leq d(x, y) + d(y, z)$ o desigualdad triangular

Las distancias más habituales son las siguientes:

- Distancia euclídea. La más común y la que se utiliza por defecto.

$$d(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

- Distancia del taxi, también conocida como distancia Manhattan debido al diseño en cuadrícula de las calles de la isla.

$$d(x, y) = \sum |x_i - y_i|$$

- Distancia de Chebyshev, también conocida como distancia del tablero de ajedrez ya que coincide con el número de movimientos que necesita el rey para moverse de una casilla a otra.

$$d(x, y) = \text{máx}\,|x_i - y_i|$$

- Distancia de Hamming, para vectores lógicos, es el número de bits que tienen que cambiarse para transformar un vector de bits en otro.

$$d(x, y) = \frac{c_{01} + c_{10}}{n}$$

donde c_{ij} es el número de ocurrencias de $x[k] = i, y[k] = j$ para $k < n$.

- Distancia de Mahalanobis. Distancia muy útil para determinar la similitud entre dos variables aleatorias multidimensionales al tener en cuenta la correlación y la escala de ellas.

$$d(x, y) = \sqrt{(x - y)V^{-1}(x - y)^T}$$

donde V es la covarianza y V^{-1} la inversa de la matriz de covarianza.

Además, otras que son ampliamnete utilizadas son Bray-Curtis, Canberra, coseno, Minkowski, o la euclídea normalizada. Para el caso de variables booleanas, además de la ya mencionada arriba distancia Hamming se han empleado: dado, Jaccard-Needham, Kulsinski, Rogers-Tanimoto, Russell-Rao, Sokal-Michener, Sokal-Sneath y Yule.

Una vez definidas la distancia a utilizar entre cada par de observaciones, se pueden definir distintos criterios de enlace entre dos clústers u y v. Los más habituales son los siguientes:

- Método *single* o del punto más cercano.

$$d(u, v) = \text{mín}\,(dist(u[i], v[j]))$$

para todos los puntos i en el cluster u y todos los j en v. Este algoritmo se centra en la separación entre clusters pero no en la cohesión de los clusters y permite formas geométricas más flexibles que en otros casos.

- Método *complete*, también conocido como Algoritmo del punto lejano o Algo-

ritmo de Voor Hees.

$$d(u,v) = \max\left(dist(u[i], v[j])\right)$$

- Método *average* o algoritmo UPGMA.

$$d(u,v) = \sum_{ij} \frac{dist(u[i], v[j])}{(|u| * |v|)}$$

- Método *weighted* o algoritmo WPGMA,

$$d(u,v) = (dist(s,v) + dist(t,v))/2 \tag{3.1}$$

donde el cluster u está compuesto por los clusters s y t.

- Método *centroid* asigna como distancia entre clusters la distancia euclídea entre sus centroides.

- Método *ward*. Según Ward la distancia entre dos clusters u y v es el incremento que se producirá en la suma del error cuadrático si se fusionan:

$$d(u,v) = \sqrt{\frac{|v|+|s|}{T}d(v,s)^2 + \frac{|v|+|t|}{T}d(v,t)^2 - \frac{|v|}{T}d(s,t)^2} \tag{3.2}$$

donde u es el nuevo cluster creado a partir de s y t y $T = |v| + |s| + |t|$. En este caso, a diferencia de K-means, el número de puntos interviene en la fórmula por lo que dados dos pares de clusters cuyos centros están distanciados por igual, el algoritmo de Ward fusionará los de menor cardinalidad.

3.3. Elección del número de clusters óptimo

Los algoritmos de clustering se utilizan principalmente de dos propósitos distintos según la problemática:

- Se conoce a priori el número de distintas clases en la población y se quieren obtener los cortes en el vector de características de las observaciones para conocer qué características tiene cada clase. Nuevas observaciones podrán ser categorizadas a partir del conocimiento extraído en el clustering.

- No se conoce cuántas clases distintas existen en la población y se quiere conocer esto a partir del clustering.

Es en el segundo caso en el que no se tiene información sobre el k a utilizar en el algoritmo K-means o el corte a aplicar en el dendrograma en clustering jerárquico. En lo que sigue se describen distintos métodos basados en heurísticas para determinar el número de clusters óptimo.

3.3.1. Coeficiente silueta

Para cada observación x se tienen dos medidas:

- Cohesión $a(x)$: grado de similitud del elemento respecto del cluster. Se obtiene como la distancia promedio de x a todos los puntos en el mismo cluster.

- Separación $b(x)$: grado de disimilitud del elemento respecto a elementos que han sido identificados en otras clases. La separación más utilizada es la distancia promedio entre x y todos los elementos del cluster más cercano, aunque también se utilizan otras medidas para valorar la separación.

El coeficiente silueta para x se define entonces como

$$s(x) = \frac{b(x) - a(x)}{\text{máx}\{a(x), b(x)\}}$$

y para todo el agrupamiento es

$$SC = \frac{1}{n} \sum_{x} s(x) \tag{3.3}$$

donde n es el número de observaciones.

Intuitivamente, un agrupamiento bien definido debería corresponderse con que para cada elemento x se tiene que $a(x) << b(x)$, es decir, x está muy cercano respecto de los elementos de su cluster en comparación con los elementos del cluster más cercano. El coeficiente $s(x)$ toma los valores en el rango $[-1, 1]$, donde -1 corresponde con una mala elección del número de clusters y 1 indica clusters bien definidos.

De esta forma, una de las técnicas que se usa con el coeficiente silueta es elegir un rango de valores para k, y elegir k de forma que el coeficiente silueta para el agrupamiento sea máximo.

3.3.2. Índice Calinski-Harabasz

Dado k, se define el índice de Calinski-Harabasz como:

$$s(k) = \frac{SS_B}{SS_W} * \frac{N-k}{k-1}$$

SS_B es la varianza entre clusters

$$SS_B = \sum_{i=1}^{k} n_i d(m_i, m)^2$$

donde m_i es el centroide del cluster i y m es la media de todas las observaciones.

SS_W es la varianza intra-cluster:

$$SS_W = \sum_{i=1}^{k} \sum_{x \in S_i} d(x, m_i)^2$$

Para que los clusters estén bien definidos deben tener valores grandes para SS_B (medida de separación) y pequeños para SS_W (medida de cohesión). De esta forma el índice de Calinski-Harabasz es una adaptación del método F-test de ANOVA, SS_B con $k-1$ y SS_W con $n-k$ grados de libertad por lo que aparecen en la fórmula pues SS_B debe ser proporcional a $k-1$ y SS_W proporcional a $n-k$.

Al igual que el coeficiente silueta, es un método que funciona generalmente bien para clusters convexos.

3.3.3. Método del codo

Bibliografía

[1] J. Lukas, J. Fridrich, M. Goljan (2006)."Digital camera identification from sensor pattern noise". *IEEE Transactions on Information Forensics and Security, 1(2), 205-214.*

[2] Daubechies wavelet. Available: `https://en.wikipedia.org/wiki/Daubechies_wavelet`

[3] K. Dabov, A. Foi, V. Katkovnik *et al* (2007). "Image denoising by sparse 3-d transform domain collaborative filtering". *IEEE Trans. Image Process., 16(8), pp. 2080-2095*

[4] F. Gisolf, A. Malgoezar, T. Baar, *et al* (2013). "Improving source camera identification using a simplified total variation based noise removal algorithm". *Digital Invest., 10(3), pp. 207-214*

[5] M. Al-Athamneh, F. Kurugollu, D. Crookes, M. Farid (2016). "Digital video source identification based on green-channel photo response non-uniformity (G-PRNU)". *Sixth International Conference on Computer Science, Engineering & Applications*

[6] T. Filler, J. Fridrich, M. Goljan (2008). "Using sensor pattern noise for camera model identification". *15th International Conference on Image Processing, San Diego, CA, 2008, pp. 1296-1299*

[7] A. Murat (2015). "Digital Video Processing, Second Edition". *Prentice Hall Signal Processing*

[8] A. Bovik (2005). "Handbook of Image and Video Processing, Second Edition". *Academic Press*

[9] M. Parker, S. Dhanani (2012). "Digital Video Processing for Engineers". *Newnes*

[10] Wiener Filter. Available: `https://en.wikipedia.org/wiki/Wiener_filter`

[11] K. Sowmya, H. Chennamma (2015). "A survey on video forgery detection". *International Journal of Computer Engineering and Applications, Volume IX*

[12] P. Bestagini, M. Fontani, S. Milani, M. Barni *et al* (2012). An overview on video forensics. *APSIPA Transactions on Signals and Information Processing, V1, pp. 1229-1233*

[13] N. Mondaini, R. Caldelli, A. Piva, M. Barni *et al* (2007). Detection of malevolant changes in digital video for forensic applications. *Proc. of SPIE, Security, Steganoprahy, and Watermarking of Multimedia Contents IX, E.J.D III and P. W. Wong, eds., vol. 6505, no. 1, SPIE, 65050T*

[14] M. Kobayashi, T. Okabe, Y. Sato (2010). Detecting forgery from static-scene video based on inconsistency in noise levels functions. *IEEE Trans. Info. Forensic Secur., 5(4). pp. 883-892*

[15] V. Conotter, J. O'Brien, H. Farid (2011). Exposing digital forgeries in ballistic motion. *IEEE Trans. Info. Forensics Secur., pp 99*

[16] J. Fridrich (1998). *Image watermarking for tamper detection. ICIP (2), pp. 404-408*

[17] J. Lukás, J. Fridrich (2003). Estimation of primary quantization matrix in double compressed jpeg images. *Proc. of DFRWS*

[18] S. Milani, M. Tagliasacchi, M. Tubaro (2012). Discriminating multiple jpeg compression using first digit features. *Proc. of the 37th Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), pp. 2253-2256*

[19] W. Wang, H. Farid (2009). Exposing digital forgeries in video by detecting double quantization. *Proc. 11th ACM Workshop on Multimedia and Security, MM&Sec '09, ACM, New York, NY, pp. 39-48*

[20] W. Wang, H. Farid (2009). Exposing Digital Forgeries in Video by Detecting Double MPEG Compression. *MM&Sec'06 Proc. of the 8th workshop on Multimedia and Security, pp. 37-47*

[21] K. Sayood (2012). Introduction to Data Compression, 4th Edition. *Morgan Kaufmann*

[22] L.J. García Villalba, A. Lucila Sandoval, J. Rosales Corripio (2015). Smartphone image clustering. *Expert Systems with Applications, 42, pp. 1927-1940*

[23] Codificación Huffman. Available: `https://en.wikipedia.org/wiki/Huffman_coding`

[24] G. Lin, J. Chang (2013). Detection of Frame Duplication Forgery in Videos based on Spatial and Temporal Analysis. *International Journal of Pattern Recognition and Artificial Intelligence*

[25] Z. Huang, F. Huang, J. Huang. Detection of double compression with the same bit rate in MPEG-2 videos. *IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP)*

[26] H. Yao, S. Song, C. Qin, Z. Tang, X. Liu (2017). Detection of Double-Compressed H.264/AVC Video Incorporating the Features of String of Data Bits and Skip Macroblocks. *Symmetry*

[27] D. Vázquez-Padín, M. Fontani, T. Bianchi, P. Comesaña *et al* (2012). Detection of video double encoding with GOP size estimation. *IEEE International Workshop on Information Forensics and Security (WIFS)*

[28] W. Wang, X. Jiang, S. Wang, T. Sun (2013). Estimation of the primary quantization parameter in MPEG videos. *Visual Communications and Image Processing (VCIP)*

[29] Y. Wu, X. Jiang, T. Sun, W. Wang (2014). Exposing video inter-frame forgery based on velocity field consistency. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, pp. 2674-2678*

[30] Grubb's test. Available: `https://en.wikipedia.org/wiki/Grubbs'_test_for_outliers`

[31] Q. Wang, Z. Li, Z. Zhang, Q. Ma (2014). Video Inter-Frame Forgery Identification Based on Consistency of Correlation Coefficients of Gray Values. *Journal of Computer and Communications, 2, pp. 51-57*

[32] Q. Wang, Z. Li, Z. Zhang, Q. Ma (2014). Video Inter-Frame Forgery Identification Based on Optical Flow Consistency. *Sensors & Transducers, 166, pp. 229-234*

[33] D. Fu, Y. Q. Shi, W. Su (2009). A generalized benfords law for jpeg coefficients and its applications in image forensics. *Proc. of SPIE, Security, Steganography and Watermarking of Multimedia Contents IX, vol. 6505, pp. 39-48*

[34] R. C. Reininger, J. D. Gibson (1983). Distributions of the two dimensional DCT coefficients for images. *IEEE Trans. On Commun., vol. COM-31, pp. 835-839*

[35] J. D. Eggerton, M. D. Srinath (1986). Statistical distribution of image DCT coefficients. *Computer and Electrical Engineering, vol. 12, pp. 137-145*

[36] D. B. Tariang, A. Roy, R. S. Chakraborty, R. Naskar (2017). Automated JPEG forgery detection with correlation based location. *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*

[37] X. Jiang, W. Wang, T. Sun, Y. Q. Shi *et al* (2013). Detection of Double Compression in MPEG-4 Videos Based on Markov Statistics. *IEEE Signal Processing Letters*

[38] W. Chen, Y. Q. Shi (2009). Detection of double MPEG compression based on first digit statistics. *Lect. Notes Comput. Sci. (IWDW 2008), vol. 5450, pp. 16-30*

[39] S. Jia, Z. Xu, H. Wang, C. Feng, T. Wang (2018). Coarse-to-fine Copy-move Forgery Detection for Video Forensics. *IEEE Access*

[40] S. Kingra, N. Aggarwal, R. D. Singh (2017). Video Inter-frame Forgery Detection Approach for Surveillance and Mobile Recorded Videos. *International Journal of Electrical and Computer Engineering (IJECE), vol. 7, no. 2, pp. 831-841*

[41] Y. Su, J. Xu, B. Dong, J. Zhang (2010). A novel source MPEG-2 video identification algorithm. *International Journal of Pattern Recognition and Artificial Intelligence, vol. 24, no. 8, pp. 1311-1328*

[42] S. Naveen, J. A. Reyaz, C. Balan (2016). Video Source Identification. *International Journal of Computer Science and Information Technologies (IJCSIT), vol. 7 (1), pp. 363-366*

[43] M. Chen, J. Fridrich, M. Goljan, J. Lukás (2007). Source Digital Camcorder Identification Using Sensor Photo Response Non-Uniformity. *Proc. SPIE 6505, Security, Steganography, and Watermarking of Multimedia Contents IX, vol. 6505, 65051G*

[44] S. Yahaya, A. TS Ho, A. A. Wahab (2012). Advanced video camera identification using conditional probability features. *Proc. of the IET Conference on Image Processing, pp. 1-5*

[45] Y. Su, J. Xu, B. Dong (2009). A source video identification algorithm based on motion vectors. *Proc. of the Second International Workshop on Computer Science and Engineering, vol. 2, pp. 312-316*

[46] Dendrogram. Available `https://en.wikipedia.org/wiki/Dendrogram`