Aplicación de la Técnica SVM en el Análisis Forense de Imágenes de Dispositivos Móviles

SISTEMAS INFORMÁTICOS

CURSO 2011-2012

Sergio Aguado Rodríguez
Pedro Luis Antona Díaz

Directores

Julián García Matesanz
Departamento de Sistemas Informáticos y Computación

Luis Javier García Villalba
Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática
Universidad Complutense de Madrid

Madrid, Julio de 2012

Autorizamos a la Universidad Complutense de Madrid difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Sergio Aguado Rodríguez

Pedro Luis Antona Díaz

Resumen

Uno de los problemas a tratar dentro del análisis forense digital es la identificación de la cámara que se ha usado para obtener una determinada imagen. Debido al aumento en el uso de teléfonos móviles con cámara integrada durante los últimos años, el trabajo está orientado a este tipo de dispositivos, proporcionando al investigador forense una herramienta específica para el análisis de este tipo de imágenes.

En este trabajo se analiza la información EXIF contenida en un archivo de imagen y se comprueba porque no es una fuente fiable a la hora de obtener la marca y modelo de la cámara. Por eso desarrollamos un algoritmo que utiliza la información de los píxeles de la imagen. Está basado en las diferencias existentes en los métodos de procesamiento de la imagen que utilizan los distintos fabricantes, como el algoritmo de interpolación del color que tiene lugar en la matriz CFA del sensor, la corrección gamma o la corrección de puntos blancos. Estas diferencias originan un conjunto de huellas en la imagen que nos permitirán diferenciar la marca y modelo de la cámara fuente. Para obtener estas huellas, extraemos un conjunto de características de las imagenes, entre las que se encuentran características de color, métricas de la calidad de la imagen y estadísticas wavelet. Para realizar la predicción de la marca y modelo de la cámara utilizamos un clasificador SVM. Siguiendo un procedimiento análogo, se desarrolla otro algoritmo que permite saber si una imagen procede de una cámara o de un escáner. Finalmente, llevamos a cabo un conjunto de experimentos que demuestran la efectividad del algoritmo implementado.

Palabras clave

Teléfonos móviles con cámara, informática forense, clasificación de imágenes, máquinas de soporte vectorial.

Abstract

One of the issues involved in digital forensics is the identification of the camera used to obtain a particular image. Due to the increase use of mobile phones with integrated camera in recent years, this work is aimed at this type of devices, providing to forensic investigator a specific tool for the analysis of images taken with mobile phones.

In this work we analyze the EXIF information contained in an image file and we explain because it is not a reliable source if we want to extract the make and model of the camera from it. So we develop an algorithm that uses information from the pixels of the image. It is based on differences in the image processing methods used by different manufacturers, such as color interpolation algorithm which takes place in the color filter array, gamma correction or white point correction. These differences originate a set of footprints in the image that allow us to differentiate the brand and model of the source camera. To obtain these footprints, we extract a feature set from the images, among which are color characteristics, image quality metrics and wavelet statistics. To predict the make and the model of camera we use an SVM classifier. Following a similar procedure, we develop another algorithm that let us know if an image comes from a camera or a scanner. Finally, we perform a series of experiments to prove the effectiveness of the implemented algorithm.

Keywords

Camera mobile phones, forensics, image classification, support vector machines.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

A pesar de la complicada situación económica global, la demanda de dispositivos móviles sigue aumentando. Se estima que en 2012 las ventas de smartphones crecerán en torno a un 39 % respecto al año anterior, quitando poco a poco ventas a los teléfonos móviles tradicionales. En total se este año se venderán 1800 millones de teléfonos, 100 millones más que el año pasado. Este incremento en ventas es debido a factores como las subvenciones por parte de los operadores, la disminución del precio medio de venta debido a la caída en los costes de los componentes, o la enorme diversidad de este tipo de dispositivos [IDG].

La gran mayoría de estos teléfonos incorporan una cámara fotográfica. La calidad de este tipo de cámaras ha aumentado de tal manera que muchas personas la utilizan como sustituto de una cámara tradicional, con la ventaja de llevar el dispositivo siempre encima. De hecho, el uso de cámaras compactas está cayendo a favor del incremento en el uso de cámaras de móvil. Desde un punto de vista forense, la gran disponibilidad de este tipo de cámaras significa un aumento de pruebas en forma de imágenes tomadas con estos dispositivos, ya que estas cámaras se pueden usar para fines delictivos, como por ejemplo el robo de información de una tarjeta de crédito o la pornografía infantil. Por tanto, la identificación y/o verificación de la cámara fuente se hace necesaria para investigaciones y propósitos legales [CSA08].

El problema de la identificación de la cámara se pueden abordar en el marco del análisis forense de imágenes. El análisis forense de imágenes es un campo emergente dedicado a determinar la fuente y la autenticidad de los objetos digitales y la posible reconstrucción del historial de manipulaciones que han sufrido dichos objetos. Un primer inconveniente a la hora de determinar la autenticidad es la facilidad con la que las imágenes pueden ser creadas, editadas y manipuladas mediante el uso de sofisticadas herramientas que aparentemente no dejan ningún rastro perceptible. Otra amenaza es la manipulación de la información que contiene la identidad del dispositivo. En este caso la cabecera de la imagen, que contiene información como la marca, el modelo, la fecha y la hora, ya no sería una fuente de información fiable. Este hecho hace que pueda haber problemas a la hora de presentar imágenes digitales como prueba en un juicio. Aun así, las herramientas forenses pueden ser diseñadas para reconocer la naturaleza y localización de la manipulación y para identificar el dispositivo que creó la imagen, así como su autenticidad.

El problema de la identificación de la cámara puede ser abordado desde varios enfoques. Uno de ellos consiste en examinar la cabecera del archivo que contiene la imagen. Por ejemplo, la cabecera EXIF (Exchangeable Image File Format) forma parte del archivo JPEG de la mayoría de las cámaras. Contiene información como el tipo de cámara digital, la localización GPS, la fecha y hora en que fue tomada la fotografía, etc. Se puede obtener el modelo y marca de la cámara a partir de estos datos. Pero esta información es altamente

vulnerable, es decir, es fácilmente modificable utilizando aplicaciones que permiten su edición de manera sencilla. Por lo tanto, la información de la cabecera puede ser alterada de manera malintencionada.

Otra aproximación a este problema se basa en las diferencias existentes en los métodos de procesamiento de la imagen que realizan los distintos modelos de cámaras. Durante esta etapa de adquisición de la imagen, las cámaras ejecutan algoritmos de demosaicing, corrección gamma, procesamiento del color, balance de blancos, compresión y almacenamiento. Estos algoritmos pueden variar de un fabricante a otro. Por lo tanto, la imagen final puede contener algunos rasgos o patrones independientemente del contenido inicial de la imagen. La idea es obtener un conjunto de características que nos permitan diferenciar entre estos patrones. Una variante de esta línea de investigación es hacer uso de las diferencias en la interpolación de la matriz CFA, un filtro que emplean la mayoría de las cámaras y que permite al sensor diferenciar entre los distintos colores. Estos filtros solo pueden detectar la intensidad de uno de los tres colores (rojo, verde y azul) por cada píxel, de manera que el resto de muestras de color se obtiene por interpolación.

El tercer enfoque a este problema es usar el patrón de ruido del sensor de la cámara. Debido a defectos en el proceso de fabricación, se introducen en el sensor patrones de ruido como la no uniformidad de los píxeles o el polvo en la lente. Estudios han demostrado que cada sensor tiene un patrón de ruido distinto, por lo que podemos utilizar estas características para identificar la fuente de adquisición de la imagen [CLW06].

Todos estos enfoques están en un principio desarrollados para cámaras tradicionales, por lo que resulta necesario facilitar al analista forense una herramienta específica centrada en torno a los dispositivos móviles tan extendidos en los últimos años.

1.1 Objeto de la Investigación

Lo que se pretende con este proyecto es abordar uno de los principales problemas que trata el análisis forense de imágenes: la identificación de la cámara que originó una determinada imagen. Más concretamente, se desarrolla un algoritmo para determinar la marca y el modelo de la cámara con la cual fue adquirida. El trabajo adapta otros estudios anteriores realizados sobre cámaras digitales a las cámaras de teléfonos móviles, ya que la popularización y el progresivo aumento de calidad de este tipo de dispositivos en los últimos años hace que una aplicación centrada en ellos sea de gran utilidad.

Para lograr dicho objetivo, debemos extraer un conjunto de características de las imágenes que hacen que podamos diferenciar unas de otras. Esta diferenciación se puede llevar a cabo partiendo de la base de que la imagen obtenida con una cámara cualquiera se ve afectada por los diferentes algoritmos que usan los fabricantes para:

- La interpolación del color en la matriz CFA (algoritmo de demosaicing).
- El procesamiento del color y otros filtros.

Como resultado de la extracción de características que debemos llevar a cabo para cada una de las bandas de la imagen (RGB), podremos observar distintos patrones independientemente del contenido de la imagen. Las características se pueden dividir en tres grupos:

1. **Características de color**: valor medio de los píxeles, correlación entre los distintos pares de las bandas RGB, centro de masa de la distribución de vecindad y la relación entre la energía de cada una de las bandas RGB.

2. **Métricas de la calidad de la imagen (IQM, *Image Quality Metrics*)**: es un conjunto de 40 características que miden la "calidad" de la imagen (luminosidad, calidad del color, nitidez, etc.).

3. Estadísticas en el dominio wavelet

Una vez que obtenemos las características de imágenes tomadas por dispositivos de diferentes marcas y modelos, debemos introducir esta información en un clasificador. Para el proyecto se ha optado por una máquina de soporte vectorial (SVM, Support Vector Machine), la cual debemos entrenar y testear con el conjunto de imágenes recolectado para que posteriormente pueda predecir la marca y modelo de una foto cualquiera tomada con alguno de los dispositivos con los que ha sido entrenada.

Por otra parte, realizando un proceso análogo, nos es posible obtener un conjunto de características basadas en correlaciones de ruido. Con ellas podemos entrenar el SVM para poder diferenciar entre imágenes procedentes de una cámara e imágenes escaneadas.

De esta manera el objetivo de este trabajo es el desarrollo de un conjunto de algoritmos que ayuden al investigador forense en la tarea de la identificación de la marca y modelo de una imagen. Con la marca nos referimos al fabricante (Samsung, Nokia, Apple, etc.) y con el modelo distinguimos entre los diferentes productos del mismo fabricante (por ejemplo Nokia 5230, 5800 o 6210). Basándonos en un procedimiento similar, también se implementa un algoritmo para la distinción entre imágenes procedentes de una cámara y de un escáner.

1.2 Estructura del Trabajo

La memoria se organiza en 8 capítulos, siendo el primero la presente introducción. El resto del trabajo se estructura como sigue.

En el capítulo 2 se explica brevemente el proceso de adquisición y procesado de una imagen de manera general para una cámara cualquiera, desde que la luz entra por el sistema de lentes hasta que la imagen se almacena en la memoria del dispositivo. Se entra con un poco más de detalle a explicar los diferentes tipos de sensores utilizados actualmente así como los filtros de color, necesarios para poder diferenciar entre los diferentes colores de la escena. Por último se habla del proceso de adquisición de imágenes en escáneres.

En el capítulo 3 se explica en qué consiste el análisis forense digital, incluyendo una explicación de los diferentes trabajos que se han realizado en el campo de técnicas y algoritmos empleados en la identificación del origen de una imagen.

En el capítulo 4 se trata la distinción entre imágenes generadas por una cámara e imágenes obtenidas de un escáner. Por último se detalla el conjunto de las características utilizadas para implementar los algoritmos tanto de identificación de marca y modelo de dispositivo móvil, como los destinados a discriminar entre imágenes de cámaras y escáneres.

En el capítulo 5 se realiza un estudio de los diferentes tipos de clasificadores: clasificadores basados en distancias, clasificadores bayesianos, redes neuronales y algoritmos de agrupamiento; poniendo especial énfasis en las máquinas de soporte vectorial (Support Vector Machine o SVM), que es la técnica utilizada en el proyecto.

En el capítulo 6 se dan detalles acerca de la implementación de los algoritmos, y las herramientas y librerías utilizadas. También se explica la utilización de métodos para la optimización y mejora del rendimiento de los diferentes algoritmos, como la ejecución de múltiples procesos simultáneos y la creación de módulos en lenguaje C.

En el capítulo 7 se realizan una serie de experimentos a partir de diversos conjuntos de imágenes utilizando los algoritmos desarrollados y se exponen los resultados.

Por último, el Capítulo 8 muestra las principales conclusiones extraídas de este trabajo así como algunas líneas futuras de trabajo.

Capítulo 2

Proceso de Adquisición y Procesado de una Imagen

Para empezar comentaremos brevemente las dos fuentes de obtención de imágenes, detallando más las partes que son útiles para la diferenciación entre estas fuentes.

2.1 Proceso de Adquisición en Cámaras Digitales

Aunque muchos de los detalles del "pipeline" de una cámara pertenecen a cada fabricante, la estructura general es la misma en todas ellas. A continuación se describen brevemente cada una de las fases del proceso de adquisición de la imagen.

A la hora de capturar una imagen es necesario medir tres o más bandas para cada píxel, lo que requiere más de un sensor, y como consecuencia se eleva el coste de la cámara. La solución más económica y extendida es la colocación de una matriz CFA delante del sensor. Puede haber mecanismos que interactúan con el sensor para determinar la exposición (tamaño de apertura, la velocidad de obturación, control de ganancia automático) y la distancia focal de la lente. Estos parámetros pueden ser determinados de manera dinámica dependiendo del contenido de la escena.

Antes del sensor también se coloca un filtro antialiasing, que se encarga de limpiar la señal antes de realizar la conversión analógica-digital. Este filtro se encarga de eliminar, antes de realizar el muestreo, de las frecuencias superiores a F/2 (siendo F la frecuencia de muestreo). Este filtro reduce el aspecto desagradable de las líneas escalonadas que aparecen en la imagen mostrando unos contornos más suaves.

El sensor (CCD o CMOS) registra la imagen convirtiendo la energía lumínica en energía eléctrica. Los datos "en crudo" obtenidos del sensor necesitan ser procesados para eliminar el ruido y otros artifacts (anomalías introducidas en las señales digitales). Uno de estos procesos es la corrección de píxeles defectuosos originados por imperfecciones en el sensor, que corrige estos píxeles mediante interpolación. Otro proceso es el balance de blancos, que permite una reproducción más fiel del color, sin que haya colores dominantes que son especialmente notables en tonos neutros como el blanco. El demosaicing (o demosaicking) es el proceso más complejo desde el punto de vista computacional y las técnicas usadas suelen ser propiedad del fabricante de la cámara. Este algoritmo utiliza los valores de los pixeles vecinos para obtener todos los canales que no han sido medidos (recordemos que el sensor en cada píxel solo detecta el canal que deja pasa la matriz CFA). Otro proceso al cual se somete la imagen es la corrección gamma, que ajusta de los valores de intensidad de la imagen. Aunque estos algoritmos aparecen en el pipeline de cualquier cámara, el

proceso exacto puede variar de un fabricante a otro, e incluso de un modelo de cámara a otro, lo cual es la base del algoritmo implementado en este trabajo.

Finalmente la imagen se comprime (en cámaras de teléfonos móviles normalmente se utiliza el algoritmo JPEG) para ahorrar espacio y se almacena en la memoria del dispositivo junto con la información de la imagen en formato EXIF [RSYD05].

En la Figura 2.1 se muestra un diagrama de bloques que representa el procesamiento de la imagen [CLW06].

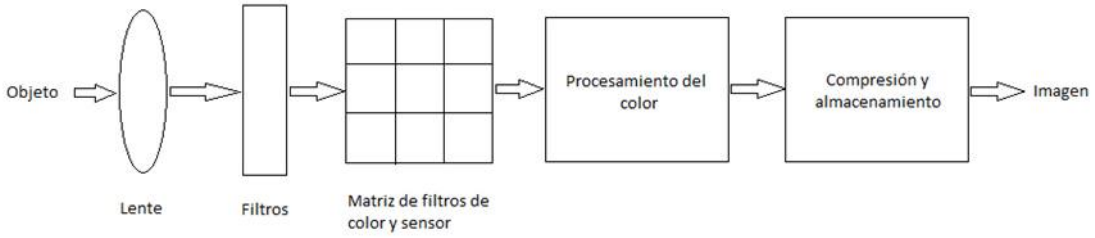


Figura 2.1: Diagrama de bloques del procesamiento de la imagen

2.1.1 Filtros de Color

El mosaico de filtros de color o matriz de filtros de color (CFA, Color Filter Array) es una de las partes más importantes en el pipeline de una cámara de un solo sensor. Está situado encima del sensor monocromo, ya sea un sensor CCD o CMOS, para adquirir la información del color de la escena. Cada celda del filtro de color deja pasar la luz de acuerdo a un rango de longitudes de onda, de tal manera que las intensidades filtradas separadas incluyen información sobre el color de la luz. De esta manera, la intensidad de la luz pasa por cada celda formando una imagen en escala de grises (en la que cada píxel se corresponde con un valor de intensidad), pero conociendo la configuración del filtro CFA se puede interpretar como una imagen a color, teniendo en cuenta que cada píxel solo contiene la intensidad de aquel color que ha dejado pasar el filtro CFA.

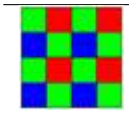
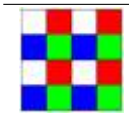
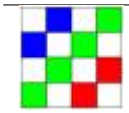
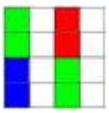
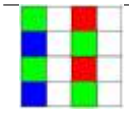
Aquí es donde actúa el proceso de demosaicing, el cual se encarga de obtener los valores que faltan para cada uno de los colores del filtro CFA mediante interpolación. El tipo de filtro de color puede variar de un fabricante a otro, por lo tanto este algoritmo también puede ser distinto. También puede haber variaciones del algoritmo en el caso de que la cámara almacene la imagen en formato CFA y el proceso de demosaicing se realice en un ordenador o por el contrario el proceso se haga directamente en la cámara.

La nitidez y la apariencia del color en los bordes y los pequeños detalles de la imagen procesada dependen del diseño del filtro CFA, por lo tanto la elección de este filtro influye en gran medida en los resultados obtenidos por la cámara. A veces, el proceso de demosaicing puede generar artifacts como el aliasing, ruido y distorsiones en el color. El uso de otro filtro puede eliminar la presencia de estas imperfecciones en determinadas áreas, a costa de degradar la calidad de la imagen en otras [LP05].

En la Tabla 2.1 se muestra una lista con diferentes tipos de filtros de color [Nak05]:

Como ejemplo de funcionamiento se muestra en la Figura 2.2 una imagen con el filtro CFA más usado, el filtro de Bayer GRGB (2 filtros verdes, 1 azul y 1 rojo) [CFA].

Tabla 2.1: Tipos de filtros de color

Imagen	**Nombre**	**Descripción**	**Tamaño del patrón (en píxeles)**
	Filtro Bayer	Es el más común. 1 píxel azul, 1 rojo y 2 verdes.	2x2
	Filtro RGBE	Como el de Bayer pero con uno de los píxeles verdes de color esmeralda, usado en algunas cámaras Sony.	2x2
	Filtro CYYM	1 píxel cian, 2 amarillos y 1 magenta. Usado en algunas cámaras Kodak.	2x2
	Filtro CYGM	C1 píxel cian, 1 amarillo, 1 verde y 1 magenta; usado en pocas cámaras.	2x2
	Filtro Bayer RGBW	Como el de Bayer pero con uno de los píxeles verdes de color blanco.	2x2
	RGBW #1	Tres ejemplos de filtros RGBW de Kodak, con el 50 % de los píxeles blancos.	4x4
	RGBW #2		4x4
	RGBW #3		2x4

Como se observa en la Figura 2.3 este filtro captura para el canal rojo el 25 % de los píxeles, para el verde el 50 % y para el azul el 25 % restante.

Esto significa que tenemos que tenemos que recuperar el 75 % de los píxeles del canal rojo, el 50 % del canal verde y otro 75 % del canal azul. El algoritmo de interpolación o reconstrucción del color (demosaicing) devuelve el resultado final (ver Figura 2.4) .

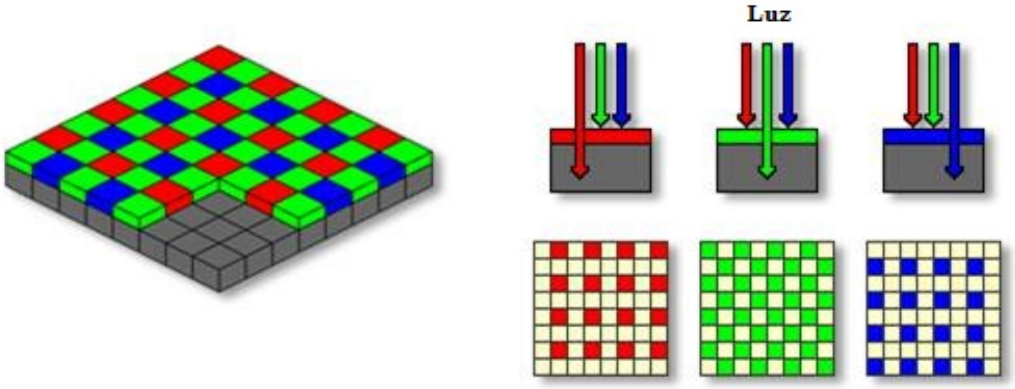


Figura 2.2: Filtro CFA

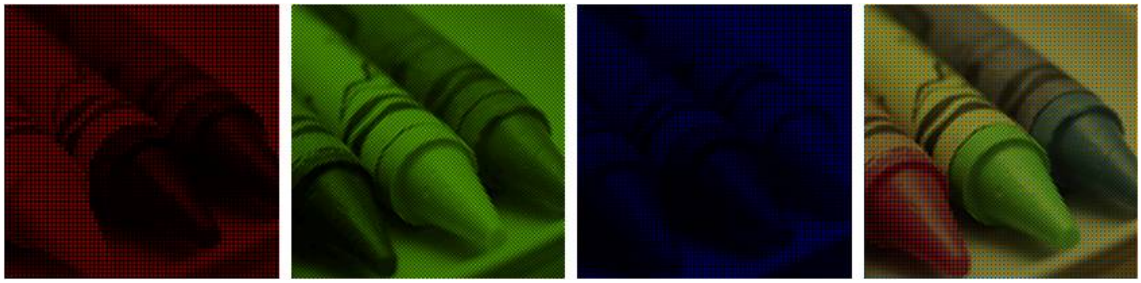


Figura 2.3: Ejemplo del Filtro CFA

Figura 2.4: Imagen Después de Aplicar el Algoritmo de Reconstrucción del color

2.1.2 Tipos de Sensor

Hoy día existen dos tipos de tecnologías utilizadas para la fabricación de sensores de cámaras digitales. Se trata de los CCD (Charge Coupled Device) y los CMOS (Complementary Metal Oxide Semiconductor). Ambos tipos de sensores están formados en su esencia por semiconductores de metal-óxido (MOS) y están distribuidos en forma de matriz.

Su función es la de acumular una carga eléctrica en cada una de las celdas de esta matriz. Estas celdas son los llamados píxeles. La carga eléctrica almacenada en cada píxel,

dependerá en todo momento de la cantidad de luz que incida sobre el mismo. Cuanta más luz incida sobre el píxel, mayor será la carga que este adquiera.

Aunque en su esencia, los CCD y CMOS funcionan de una manera muy similar, hay algunas características que diferencian ambas tecnologías.

Sensores CCD

En el caso del CCD, éste convierte las cargas de las celdas de la matriz en voltajes y entrega una señal analógica en la salida, que será posteriormente digitalizada por la cámara. En los sensores CCD, se hace una lectura de cada uno de los valores correspondientes a cada una de las celdas. Entonces, es esta información la que un convertidor analógico-digital traduce en forma de datos. En este caso, la estructura interna del sensor es muy simple, pero tenemos como inconveniente la necesidad de un chip adicional que se encargue del tratamiento de la información proporcionada por el sensor, lo que se traduce en un gasto mayor y equipos más grandes.

En el aspecto del rango dinámico, es el sensor CCD el ganador absoluto, pues supera al CMOS en un rango de dos. El rango dinámico es el coeficiente entre la saturación de los píxeles y el umbral por debajo del cual no captan señal. En este caso el CCD, al ser menos sensible, los extremos de luz los tolera mucho mejor.

En cuanto al ruido, también son superiores a los CMOS. Esto es debido a que el procesado de la señal se lleva a cabo en un chip externo, el cual puede optimizarse mejor para realizar esta función. En cambio, en el CMOS, al realizarse todo el proceso de la señal dentro del mismo sensor, los resultados serán peores, pues hay menos espacio para colocar los foto-diodos encargados de recoger la luz (ver Figura 2.5).

La respuesta uniforme es el resultado que se espera de un píxel sometido al mismo nivel de excitación que los demás, y que éste no presente cambios apreciables en la señal obtenida. En este aspecto, el que un sensor CMOS esté constituido por píxeles individuales, le hace más propenso a sufrir fallos. En el CCD, al ser toda la matriz de píxeles uniforme, tiene un mejor comportamiento. A pesar de todo, la adición de circuitos con realimentación nos permite subsanar este problema en los CMOS, los CCD siguen estando un poco por encima.

Sensores CMOS

En el caso del CMOS, aquí cada celda es independiente. La diferencia principal es que aquí la digitalización de los píxeles se realiza internamente en unos transistores que lleva cada celda, por lo que todo el trabajo se lleva a cabo dentro del sensor y no se hace necesario un chip externo encargado de esta función. Con esto conseguimos reducir costes y equipos más pequeños.

Además de ofrecernos más calidad, los CMOS son más baratos de fabricar precisamente por lo comentado arriba. Otra de las grandes ventajas es que los sensores CMOS son más sensibles a la luz, por lo que en condiciones pobres de iluminación se comportan mucho mejor. Esto se debe principalmente a que los amplificadores de señal se encuentran en la propia celda, por lo que hay un menor consumo a igualdad de alimentación. Todo lo contrario que ocurría en los CCD.

En cuanto a la velocidad, el CMOS es claramente superior al CCD debido a que todo el procesado se realiza dentro del propio sensor, ofreciendo mayor velocidad. Es esta una de las principales razones por las que Casio empezó a imponer los sensores CMOS en sus cámaras y por la cual éstas permiten grabar vídeos a velocidades de hasta 1000 fps (ver Figura 2.6).

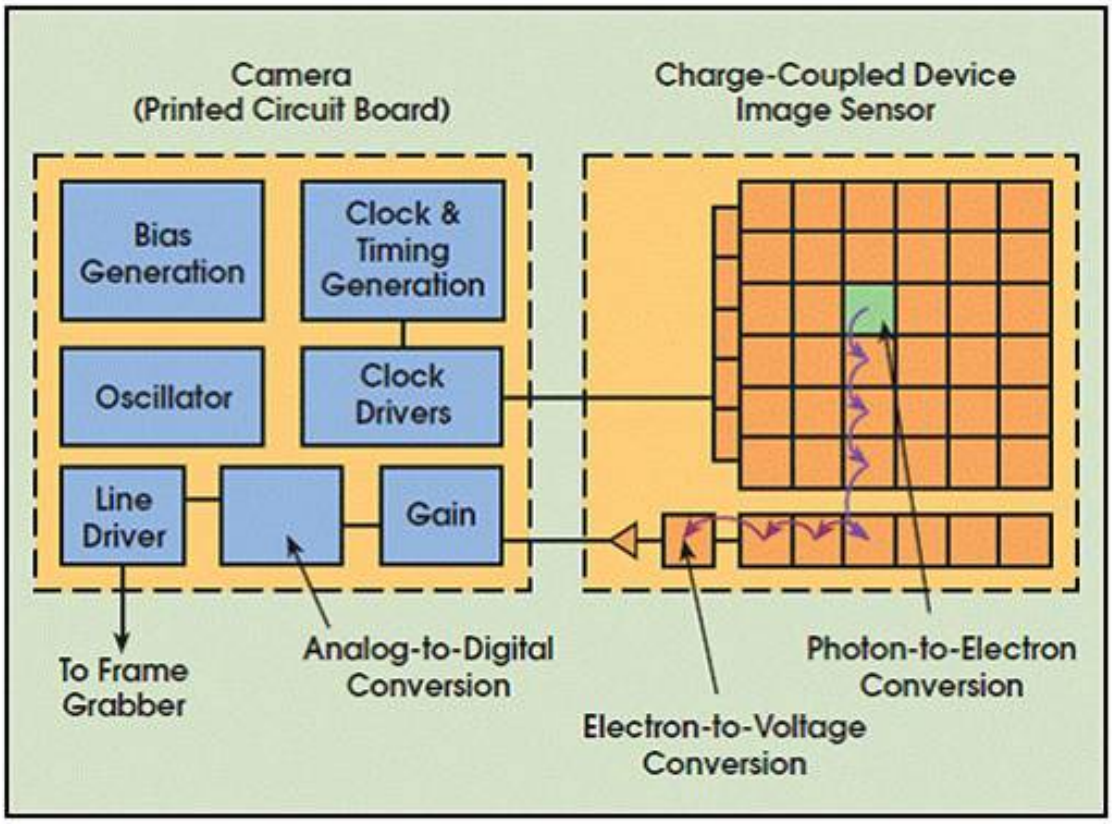


Figura 2.5: Diagrama de un sensor CCD [BIC]

Otro aspecto en el que los sensores CMOS son superiores a los CCD es en el blooming. Este fenómeno se produce cuando un píxel se satura por la luz que incide sobre él y a continuación empieza a saturar a los que están a su alrededor. Aunque este defecto puede subsanarse gracias a algunos trucos en la construcción, en el caso de los CMOS no existe ese problema.

Los sensores CMOS en sus inicios eran algo peores que los CCD, pero hoy día es un mal que está prácticamente subsanado. La tecnología CCD ha llegado a su límite y es ahora cuando se está desarrollando la CMOS. [CCD]

2.2 Proceso de Adquisición en Escáneres

Los escáneres constan de un "pipeline" muy diferente al de las cámaras fotográficas, como muestra la Figura 2.7.

Tenemos una lámpara utilizada para iluminar el propio documento que queremos digitalizar. Puede ser CCFL (*cold cathode fluorescent lamp*), una lámpara de xenón y en los viejos escáneres se usan lámparas fluorescentes normales. Otra parte importante es una barra de estabilización, que junto con un motor eléctrico consiguen mover la cabeza lectora de manera eficiente y sin tener desviaciones a la hora de leer el documento. Esta cabeza lectora tiene lentes, filtros, espejos y sensores de imagen. Por último, estos sensores de imagen suelen ser CCD (*charge-coupled devide*), aunque también nos podemos encontrar con CMOS (*complementary metal oxide semiconductor*), CIS (*contact Image Sensors*) o

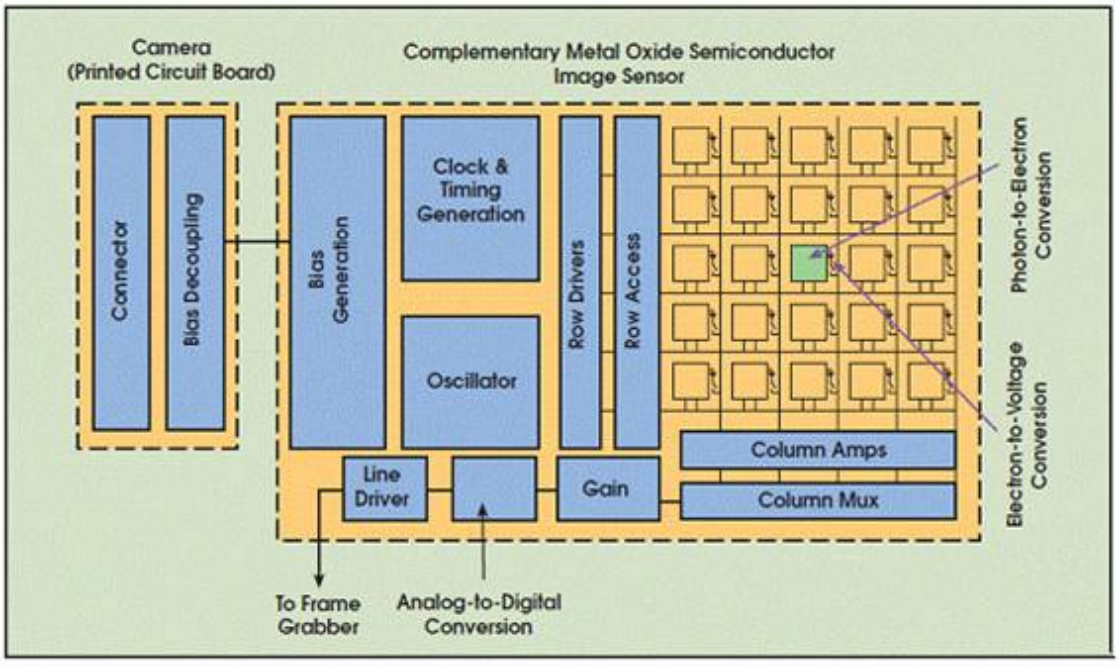


Figura 2.6: Diagrama de un sensor CMOS [BIC]

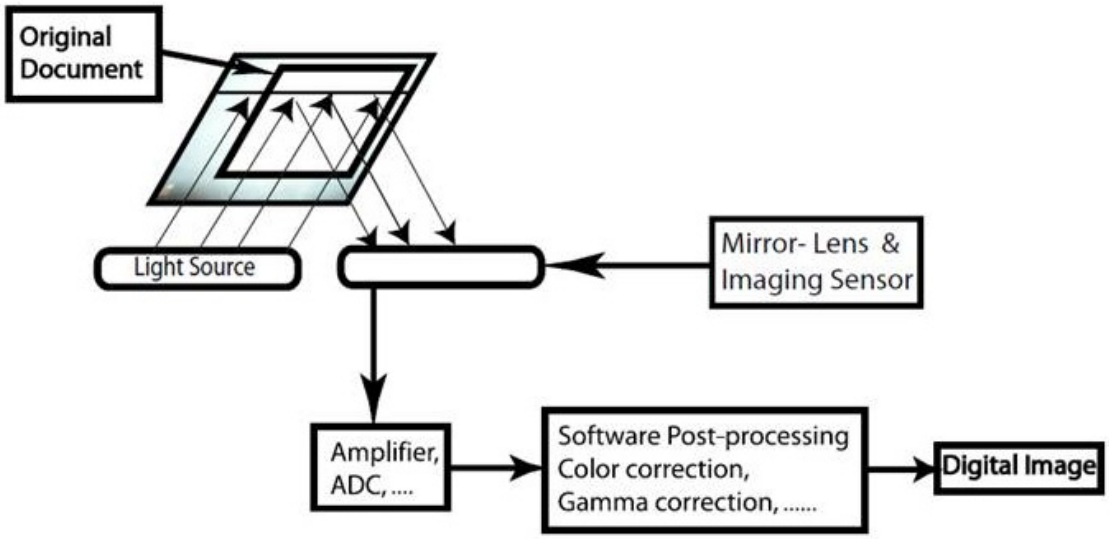


Figura 2.7: Plano de canalización de imagenes en un escaner

PMTs (*photomultiplier uve*).

Capítulo 3

Técnicas de Análisis Forense en Imágenes

Existen numerosos estudios acerca de la identificación de cámaras. Estos estudios se pueden dividir en dos grandes grupos de acuerdo a la fuente de la información que usan. Los métodos del primer grupo utilizan el ruido del sensor e imperfecciones en la matriz CCD. El segundo grupo se basa en los *artifacts* (imperfecciones en la imagen) que tienen lugar tras el algoritmo de *demosaicing* que se lleva a cabo durante el procesamiento de la imagen RAW (imagen en "crudo", que es la que genera el sensor y a la que todavía no se le ha aplicado ningún procesamiento). Finalmente se describen otros dos métodos alternativos que usan la distorsión radial de la lente y las características de polvo del sensor.

3.1 Identificación de la Cámara Basado en la Información EXIF

Para la identificación de las características de la imagen, podríamos fijarnos en los metadatos que esta guarda, también denominado EXIF. Estos metadatos siguen un estándar, el cual deberían de cumplir todas las fotografías realizadas con todos los modelos.

Se hizo un estudio previo para comprobar que tal se seguía este estándar por las diferentes marcas y los terminales dentro de estás, así como los tag con más errores. Los resultados fueron los siguientes.

En el análisis por marca se dieron los resultados mostrados en la Figura 3.1.

En cuanto al análisis por modelo vemos los datos mostrados en la Figura 3.2. Hacemos un estudio de los terminales que incumplen en mayor medida el estándar EXIF.

Los anteriores porcentajes han sido sacados de los 15 terminales con más errores y que al menos se tenía 20 fotos realizadas con ellos en la base de datos. Los tags (campos de los metadatos) más erróneos entre la base de fotografías utilizada se muestran en la Figura 3.3.

Vemos que el tag más propenso a tener errores es "FlashPixVersion" en hexadecimal 0xA000 con 772 errores en el total de las fotos. Este tag es obligatorio en cualquiera de los tipos de compresión de JPEG. También, el tag GainControl, posee un alto número de errores (536). Los demás tags vemos que son ligeramente inferiores sus apariciones con errores. Destacar por último los tags que muestran errores en el GPS, "GPSVersionID", "GPSLatitude" y "GPSLongitude", los cuales aparecen en la estadística como unos de los más problemáticos. Con todo esto vemos que el análisis de las fotos por este método

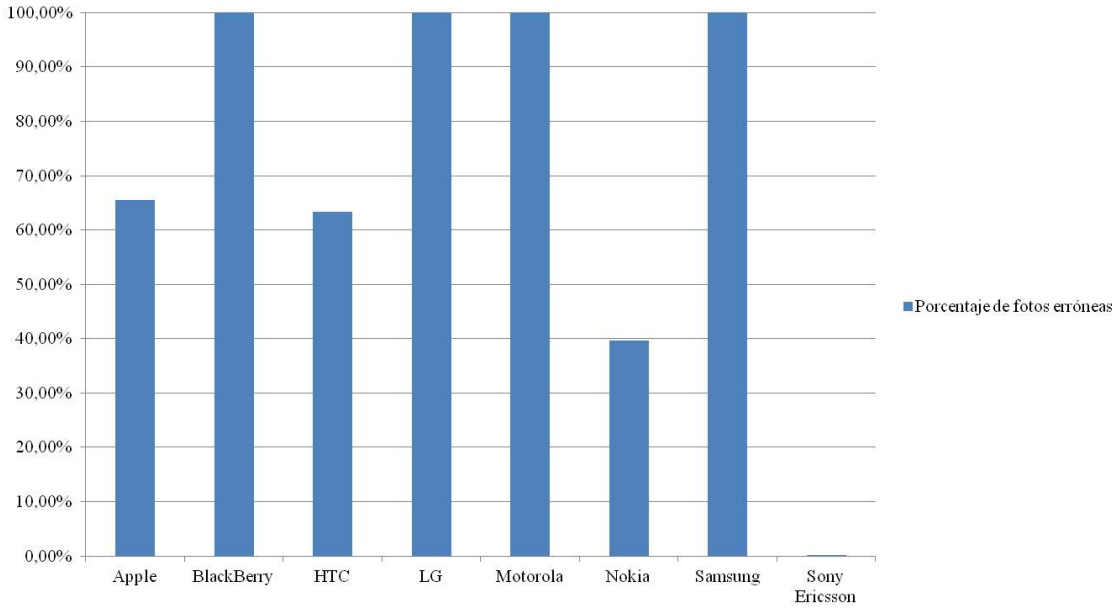


Figura 3.1: Porcentaje de fotos erróneas

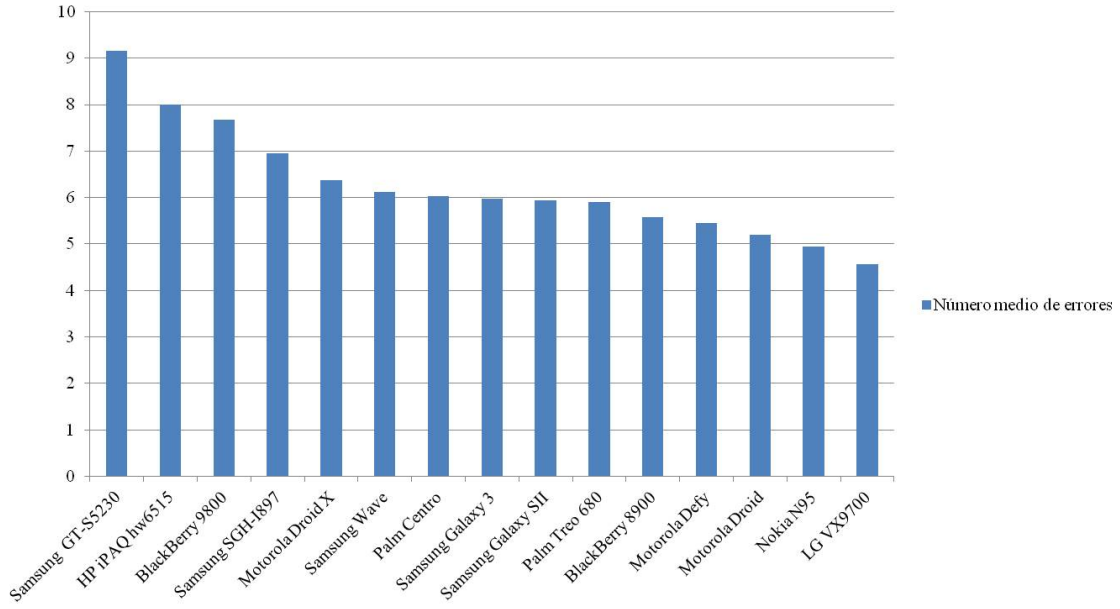


Figura 3.2: Número medio de errores

no es suficiente, en primera instancia, por los errores propios de los modelos, determinadas marcas o determinados tags. También por la enorme facilidad de ser manipulados estos datos externamente, con un sencillo programa, pudiendo cambiar las características fundamentales de la fotografía en cuestión.

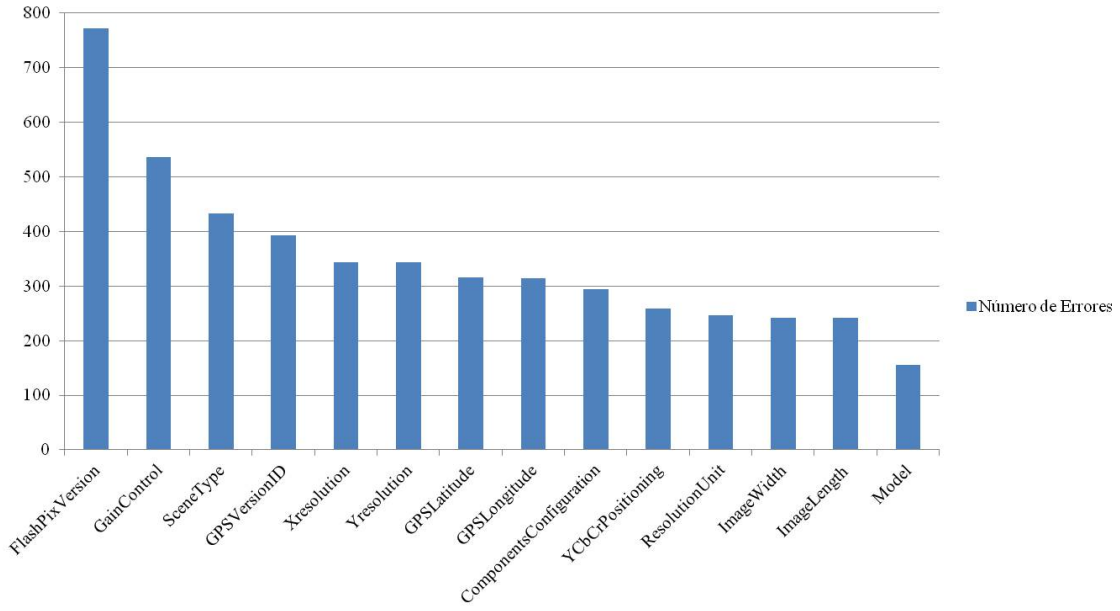


Figura 3.3: Número de errores por Etiqueta

3.2 Identificación de la Cámara Basado en el Ruido del Sensor

Los estudios de Geradts et al.[GBK+01] se centraban en evaluar diferentes características para examinar las imágenes y que permitían averiguar el origen de la cámara:

- Defectos del sensor CCD.
- El formato de los archivos usado.
- Ruido introducido en la imagen.
- Marcas de agua introducidas por el fabricante de la cámara.

Entre los defectos del sensor CCD se encuentran los puntos calientes, que son píxeles con valores muy altos para el voltaje de salida; píxeles muertos, que son píxeles con una baja capacidad de respuesta; y los píxeles trampa, que surgen de un problema en el proceso de transferencia de la carga. En la mayoría de las cámaras existe hardware para compensar este tipo de errores.

Los resultados demostraban que en las cámaras más baratas todos estos defectos eran más visibles. Las más caras tenían sensores CCD con menos errores. El nivel de compresión también afectaba a estos valores.

Lukas et al. [LFG06] realizó un estudio que analizaba el patrón de ruido del sensor de un conjunto de cámaras, el cual funciona como una huella dactilar, permitiendo la identificación única de cada cámara. Para obtener este patrón se realiza un promedio del ruido obtenido a partir de diferentes imágenes utilizando un filtro de eliminación de ruido. Para identificar la cámara a partir de una imagen dada, se considera el patrón de referencia como una marca de agua cuya presencia en la imagen es establecida mediante un detector de correlación. El estudio se realizó con 320 imágenes procedentes de 9 modelos

distintos de cámaras. También demuestran que este método está afectado por algoritmos de procesamiento de la imagen como la compresión JPEG o la corrección gamma.

Kurosawa et al. [KKS99] propusieron un nuevo método relacionado con los patrones extraídos del sensor CCD aplicado a identificar los vídeos procedentes de una determinada cámara de video. Para ello utilizaban la respuesta no uniforme (PRNU, Photo-response non-uniformity) de los sensores de imagen como característica única de la cámara. El PRNU puede ser extraído promediando los residuos de ruido de varias imágenes. Este estudio está destinado como una herramienta para luchar contra la piratería de películas.

3.3 Identificación de la Cámara Basado en los Demosaicing Artifacts

Las cámaras comerciales usan un mosaico de filtros de color simple (CFA, Color Filter Array), en lugar de tener filtros separados para cada componente de color. Los distintos modelos de cámara utilizan sus propios algoritmos de interpolación para recuperar los valores de color perdidos. Este proceso de interpolación deja huellas, lo que es la base para estos trabajos.

Mehdi et al. [MSM04] intentó capturar las diferencias en la configuración del filtro CFA y del pipeline de procesamiento de color mediante una aproximación basada en características. Estos procesos, entre los que se incluyen la interpolación del color (demosaicing), la corrección de puntos blancos o la corrección gamma, varían de un fabricante a otro e incluso entre diferentes modelos del mismo fabricante. Basándose en esto, proponen un conjunto de características que se pueden clasificar en dos clases:

- Características del color:
  - Valor medio del píxel (3 características).
  - Correlación de los pares RGB (3 características).
  - Distribución de la vecindad de centro de masa (3 características).
  - Relación entre la energía de los pares RGB (3 características).
  - Estadísticas en el dominio wavelet (9 características).
- Métricas de calidad de imagen (IQM, Image Quality Metrics). 13 características repartidas en tres clases:
  - Medidas basadas en la diferencia del valor de los píxeles.
  - Medidas basadas en correlación.
  - Medidas basadas en la distancia espectral.

Este conjunto de características junto con algunas IQM adicionales extraídas del trabajo de Celiktutan et al. [CSA08] son las usadas en este proyecto para la parte de identificación de la marca y modelo de teléfono móvil. Dichas características serán explicadas más detenidamente en el capítulo 4.

Una vez obtenidas las características se utiliza un clasificador SVM (Support Vector Machine) para el entrenamiento, testeo y predicción de las imágenes. Un clasificador de este tipo también es usado en el proyecto.

Long et al. [KKS99] usó correlaciones entre píxeles. Definieron un modelo de correlación cuadrática de los pixeles y obtenían una matriz de coeficientes para cada banda de color. Utiliza redes neuronales para la clasificación.

Swaminathan et al. [SWL07] investigaron los artifacts originados por el proceso de demosaicing usando el método de análisis por síntesis. Dividían la imagen en tres regiones basadas en características del gradiente y luego estimaban los coeficientes de interpolación a través de la descomposición de valores singulares (SVD) para cada región y banda de color por separado. Luego re-interpolaban el patrón CFA y elegían uno que minimizara la diferencia entre la imagen final estimada y la imagen producida por la cámara.

3.4 Identificación de la Cámara por Métodos Alternativos

Choi et al. [CLW06] propusieron una alternativa a los métodos de clasificación anteriores que usa la distorsión radial de las lentes. Todas las lentes producen algún tipo de aberración que deja huellas únicas en cada una de las imágenes capturadas. El grado de distorsión radial de cada imagen puede ser medido en forma de 2 parámetros. El procedimiento para su obtención consta de las siguientes fases:

1. Detección de bordes.
2. Extracción de segmentos distorsionados.
3. Medición del error de la distorsión.

Estos parámetros, junto con las características propuestas por Mehdi et al. (apartado anterior), forma un nuevo vector de 36 características. Posteriormente se evalúa la tasa de acierto mediante un clasificador SVM. Los resultados de este trabajo demuestran que la inclusión de estas características mejoran los resultados que habían sido obtenidos sin ellas.

Dirik et al. [DSM07] utilizó las características de polvo del sensor en la identificación de imágenes. El polvo en el sensor es un problema asociado a las cámaras réflex digitales (DSLR, Digital single lens reflex). Este problema surge debido a las partículas de polvo que son atraídas al sensor cuando se cambia la lente, creando un patrón de polvo en frente del sensor de imagen. Este polvo aparece en la imagen en forma de artifacts, los cuales son más visibles a valores menores de apertura. Este patrón aparecerá en las imágenes hasta que la superficie del sensor sea limpiada, lo que permite identificar qué cámara DSLR concreta generó una determinada imagen. Los resultados obtenidos muestran que este método tiene una tasa muy baja de falsos positivos.

La Tabla 3.1 resume las principales propiedades de los diferentes algoritmos de identificación de cámaras.

Tabla 3.1: Principales propiedades de los diferentes algoritmos de identificación de cámaras

Método	**Propiedades**	**Comentarios**
Interpolación CFA	Trata las partes lisas y no lisas de manera separada, y usa coeficientes de interpolación como características junto con un clasificador SVM.	Le afecta la compresión JPEG.
Ruido del sensor	Construye un patrón de ruido promediando los múltiple residuos de ruido obtenidos mediante un filtro wavelet de eliminación de ruido, y determina la presencia del patrón utilizando un predictor de correlación.	El residuo del ruido está afectado por el contenido de la imagen, y debe ser extraído de las regiones no saturadas de la imagen. Las transformaciones geométricas como la reducción de la resolución o los recortes en la imagen deforman el patrón de ruido.
Distorsión radial de la lente	Estima los parámetros de distorsión radial de la lente y los usa en un clasificador SVM.	Está influido por la distancia focal de la lente.
Características de polvo del sensor	Usa características de polvo en cámaras DSLR (Digital single lens reflex).	Este tipo de características son difícilmente detectables en regiones complejas de la imagen.
Método de interpolación CFA	Construye un espacio de búsqueda de patrones CFA, estima los coeficientes de interpolación de color mediante aproximación lineal y encuentra un patrón CFA que produce el mínimo número de errores de interpolación.	Existe un número limitado de algoritmos de interpolación y patrones CFA usados por los fabricantes. El método de interpolación no difiere mucho entre fabricantes.
Demosaiking artifacts	Usa un clasificador SVM con un vector de características compuesto por características del color, estadísticas wavelet y métricas de la calidad de la imagen.	Realizando una selección de estas características o una fusión con otras técnicas se puede mejorar la precisión de la clasificación.
Modelo de correlación entre píxeles	Modela la correlación periódica entre píxeles de forma cuadrática y utiliza redes neuronales para la clasificación.	Es sensible a operaciones que modifican la correlación de vecindad.

Capítulo 4

Algoritmos de Clasificación de la Fuente de una Imagen

4.1 Distinción entre las Imágenes de Cámara e Imágenes Escaneadas

4.1.1 Ruido en las Imágenes

El proceso de generación de imágenes suele introducir varios defectos en estas, los cuáles crearán ruido que aparecerá en la imagen final. Un tipo de ruido es causado por defectos de array, esto incluye hot point defects, dead pixels, pixel traps, column defects y cluster defects. Esto causa que dichos pixeles difieran en gran medida de la imagen original, siendo en muchos casos indiferente que se tenga una u otra imagen, ya que este pixel mostrará siempre el mismo valor. Por ejemplo los dead pixels(pixeles muertos) aparecerán en la imagen como un pixel negro en la imagen resultante, los hot point pixels aparecerán muy brillantes siempre. El patrón de ruido en una imagen se refiere a cualquier patrón espacial que no cambia de una imagen a otra y es causado por un "*dark current*" y un PRNU (photoresponse nonuniformity) [KMC+06].

Tenemos filtros para conseguir suavizar el efecto de este ruido. Por sencillez, velocidad y facilidad de implementación nosotros usaremos el Filtro Gaussiano. Este filtro será usado posteriormente para eliminar de manera rápida y efectiva el ruido en las imágenes y con estos datos realizar diferentes operaciones que nos llevarán a determinar las diferentes características.

4.1.2 Descripción del Procedimiento para Identificar la Fuente

Nuestro objetivo es conseguir una serie de características de las imágenes que nos permitan diferenciar claramente un escáner y una cámara. Nos basaremos, como hemos comentado antes, en la forma lineal de los escáneres o matricial de las cámaras de reconocer la imagen para distinguirlas. Sea una imagen inicial de M*N pixeles, siendo M las filas y N las columnas. Ahora denotamos I_{noise} el ruido correspondiente a la imagen original y $I_{denoised}$ a la imagen sin ruido. Entonces tenemos:

$$I_{noise} = I - I_{denoised}$$

Para conseguir la imagen sin ruido utilizaremos el filtro gaussiano, el cual nos da la velocidad necesaria para conseguir analizar una gran cantidad de fotos en poco tiempo.

Después restaremos cada componente de color (RGB) a la imagen original, lo cual nos dará la componente de ruido de cada pixel desglosado en los 3 colores.

El ruido de la imagen original I_{noise} puede ser modelado como la suma de dos componentes, el ruido constante $I_{noiseconstant}$ y el ruido aleatorio $I_{noiserandom}$. Para los escáneres el ruido constante solo depende del índice de la columna ya que el mismo sensor es trasladado verticalmente para generar la imagen completa. La media del ruido de todas las columnas puede ser usada como patrón de referencia $\hat{I}_{noiseconstant}(1,j)$ ya que las componentes aleatorias del ruido se anularan.

$$\hat{I}_{noiseconstant}(1,j) = \frac{\sum_{i=1}^{M} I_{noise}(i,j)}{M}, 1 \leq j \leq N$$

Para detectar la similitud entre las diferentes filas con el patrón de referencia, usaremos la correlación de estas con dicho patrón.

$$correlation(X,Y) = \frac{(X-\overline{X})\cdot(Y-\overline{Y})}{\|X-\overline{X}\|\cdot\|Y-\overline{Y}\|}$$

Haremos lo mismo para las columnas.

Después de tener las correlaciones de las filas y de las columnas pasaremos a la obtención de las características en sí.

Es importante tener en cuenta, a la hora de obtener las características, que la orientación de la foto de entrada es fundamental y que si no sabemos si una imagen esta girada o no debemos descartarla ya que la imagen girada nos cambiará todas las características.

4.1.3 Características de Primer Orden y Alto Orden

Estas características son las que obtendremos para la distinción de imágenes por patrones de correlación sobre filas y columnas, orientado principalmente a imágenes escaneadas contra imágenes de cámara fotográfica.

Para cada tipo de correlación (filas o columnas) obtendremos estadísticos de primer orden los cuales son: media, mediana, máximo y mínimo. En el documento original estas características incluían también la moda, pero después de varios análisis y experimentos se vio como una característica inútil, ya que al tratarse de valores flotantes, no había valores repetidos nunca y este valor siempre tomaba el de inicio. Se probó a truncar los valores flotantes hasta determinados decimales, pero el resultado no era bueno, disminuyendo incluso el porcentaje de aciertos del clasificador.

Luego las características de orden alto son varianza, kurtosis y skewness. Todas ellas miden valores estadísticos más específicos que las anteriores. A todo esto le añadimos el ratio entre las correlaciones de filas y de columnas.

A todas estás, se vio oportuno añadir una característica nueva, basada también en el ruido de la imagen. La nueva característica medía el ruido medio por pixel. Esta característica no dependía por tanto de las correlaciones de filas o columnas con el patrón de referencia, sino que era independiente y nos habría un camino a distinguir, ya no estos tipos de imagen, sino a otros grupos, como pueden ser las imágenes generadas por computador.

En total tenemos 7 características de filas, 7 de columnas, el ratio y el ruido medio del pixel, lo que hacen un total de 16 características obtenidas por el algoritmo de distinción de imágenes por correlación sobre patrones de ruido de filas y columnas.

4.2 Extracción de Características para la Identificación de la Marca y Modelo del Dispositivo

Uno de los principales problemas en el análisis forense de imágenes es la identificación de la cámara utilizada para obtener una imagen. En este documento se proponen una serie de técnicas que nos ayudarán en la identificación de la marca y el modelo de dicha cámara. La idea es obtener un conjunto de características a partir de una imagen que estén asociadas a un modelo de cámara en concreto. Las características de las imágenes dependen en gran medida de:

- La configuración de los filtros CFA (color filter arrays) y el algoritmo de demosaicing (algoritmo mediante el cual se calculan las componentes de color perdidas en los distintos píxeles del sensor).
- El procesamiento del color.

Como resultado de estos dos elementos el procesamiento de las señales contenidas en la bandas RGB pueden contener tratamientos y patrones específicos. Con el objetivo de determinar las diferencias en las características del color para los diferentes modelos de cámaras es necesario examinar las estadísticas de primer y segundo orden de las imágenes tomadas con ellas. A continuación se proponen como candidatas un conjunto de características que nos ayudarán en la tarea de clasificar las imágenes.

4.2.1 Características de Color

- **Valor medio de los píxeles**: para esta medida se asume que el promedio de los valores de los canales RGB de una imagen debe dar como resultado el color gris, siempre y cuando la imagen tenga suficientes variaciones de color. Esta medida se realiza para cada uno de los canales RGB (3 características).
- **Correlación de los pares RGB**: con esta medida se expresa el hecho de que dependiendo de la estructura de la cámara, la correlación entre las diferentes bandas de color puede variar. En la implementación de esta característica utilizamos el coeficiente correlación de Pearson para determinar la correlación en los valores de cada una de las bandas. Como resultado obtenemos 3 características que provienen de medir la correlación entre las bandas RG, RB y GB.
- **Distribución de vecindad del centro de masa**: esta medida se calcula para cada banda por separado. Primero se calcula el número total de píxeles para cada valor (resultando un vector de 256 componentes). Después, con estos valores calculados se obtienen la suma de los valores vecinos, es decir, para cada valor i del vector anteriormente calculado se suma la componente i-1 y i+1. Por último se calcula el centro de masa de este último vector, lo que va a devolver un valor entre 0 y 255. (3 características).
- **Ratio de la energía entre los pares RGB**: esta característica depende del proceso de corrección de puntos blancos que realiza la cámara. Son 3 características que están definidas como:

$$E_1 = \frac{|G|^2}{|B|^2} \; E_2 = \frac{|G|^2}{|R|^2} \; E_3 = \frac{|B|^2}{|R|^2}$$

4.2.2 *Image Quality Metrics* (IQM)

Los diferentes modelos de cámara producen imágenes de diferente calidad. Puede haber diferencias en la luminosidad de la imagen, la nitidez o en la calidad del color. Estas diferencias que se observan al comparar fotos de varias cámaras hacen que se proponga un conjunto de métricas de calidad (Image Quality Metrics o IQM) como características que nos ayudan a diferenciar la fuente de las imágenes. Existen diferentes categorías para las IQM's:

- Medidas basadas en las diferencia de los píxeles.
- Medidas basadas en la correlación.
- Medidas basadas en la distancia espectral.

La Tabla 4.1 muestra un conjunto de métricas de calidad.

Tabla 4.1: Conjunto de Image Quality Metrics

Minkowsky Metric $\gamma = 1$	Distancia Czekonowsky
Minkowsky Metric $\gamma = 2$	Spectral Phase
Normalized Cross Correlation	Spectral Magnitude
Structural content	Weighted Spectral Distance
Normalized absolute error (HVS)	Median Block Spectral Magnitude
HVS based L2	Median Block Spectral Phase
Laplacian Mean Square Error	Median Block Weighted Spectral Distance

Para obtener este conjunto de métricas es necesario además de la imagen original, una imagen filtrada con la que poder realizar los diferentes cálculos.

Filtro de Suavizado Gaussiano

Para obtener una imagen de referencia con la cual calcular las diferentes métricas se realiza un filtrado para reducir el ruido de la imagen original. En la implementación utilizamos un filtrado gaussiano para llevar a cabo el suavizado de la imagen.

Para obtener el núcleo gaussiano bidimensional se utiliza la siguiente fórmula:

$$\left(2\pi\sigma^2\right)^{-1} * e^{\frac{-\left(i^2+j^2\right)}{2\sigma^2}}$$

Una vez obtenido el núcleo se normaliza para que la suma de todas sus componentes sea 1. Esto es necesario para obtener una imagen suavizada pero con los mismos colores que la original. La normalización se realiza dividiendo cada componente entre la suma de los valores de todas las componentes.

Para obtener las métricas se utiliza un filtro con un núcleo de tamaño 3x3 con $\sigma = 0.5$:

$$h = \begin{vmatrix} 0.01134 & 0.08381 & 0.01134 \\ 0.08381 & 0.61934 & 0.08381 \\ 0.01134 & 0.08381 & 0.01134 \end{vmatrix}$$

Cada píxel de la nueva imagen se obtiene realizando la transformación de vecindad sobre el píxel de la imagen original utilizando el núcleo anteriormente calculado:

$$I'(x,y) = h(0,0) * I(x-1,y-1) + h(0,1) * I(x,y-1) + h(0,2) * I(x+1,y-1) + h(1,0) * I(x-1,y) + h(1,1) * I(x,y) + h(1,2) * I(x+1,y) h(2,0) * I(x-1,y+1) + h(2,1) * I(x,y+1) + h(2,2) * I(x+1,y+1)$$

En los bordes de la imagen es necesario tener cuidado al hacer la transformación, en nuestro caso se ha optado por considerar un borde exterior con píxeles de valor 0.

Distancia Czekonowsky

La distancia Czekonowsky es una métrica útil para comparar vectores con componentes no negativas como es el caso de las imágenes en color:

$$M = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(1 - \frac{2 \sum_{k=1}^{3} min\left(C_k(i,j), \hat{C}_k(i,j) \right)}{\sum_{k=1}^{3} \left(C_k(i,j) + \hat{C}_k(i,j) \right)} \right)$$

Métricas de Minkowsky

Las métricas de Minkowsky para $\gamma = 1$ y $\gamma = 2$ se basan en la siguiente fórmula:

$$M_\gamma = \frac{1}{K} \sum_{k=1}^{K} \left\{ \frac{1}{N^2} \sum_{i,j=1}^{N} \left| C_k(i,j) - \hat{C}_k(i,j) \right|^\gamma \right\}^{\frac{1}{\gamma}}$$

que calcula la norma L_γ de disimilitud entre dos imágenes. K se refiere a cada uno de los canales de la imagen y N^2 define el número total de píxeles. Hay que tener en cuenta que esta fórmula realiza el promedio de la métrica de Minkowsky de cada canal.

$\gamma = 1$ se corresponde con el *Mean Absolute Error* (MAE) y $\gamma = 2$ con el *Mean Square Error* (MSE). En ambos casos, valores elevados de MAE o MSE se corresponden con imágenes de baja calidad.

Mean Absolute Error

$$MAE = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left| C_k(i,j) - \hat{C}_k(i,j) \right|$$

Mean Square Error

$$MSE = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left(C_k(i,j) - \hat{C}_k(i,j) \right)^2$$

$C_k(i,j)$ y $\hat{C}(i,j)$ se refieren a los píxeles en la posición (m,n) de la imagen original y la imagen suavizada respectivamente. MxN es el tamaño de la imagen. Esto se aplica a cada una de las bandas por separado, por lo que tenemos tres características para el MAE y otras tres para el MSE.

***Laplacian Mean Square Error* (LMSE)**

Está basada en la importancia de la medición de los bordes. Un valor alto del LMSE indica que la imagen es de baja calidad. Se define de la siguiente forma:

$$LMSE = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} \left[L(x(m,n)) - L\left(x^{(m,n)} \right) \right]^2}{\sum_{m=1}^{M} \sum_{n=1}^{N} \left[L(x(m,n)) \right]^2}$$

donde $L(x(m,n))$ es el operador Laplaciano:

$$L(x(m,n)) = x(m+1,n) + x(m-1,n) + x(m,n+1) + x(m,n-1) - 4x(m,n)$$

Normalized Cross Correlation

La cercanía entre dos imágenes digitales también puede ser cuantificada en términos de una función de correlación. La medida de correlación cruzada normalizada está definida para cada banda de la imagen (k) como:

$$NCC = \frac{\sum_{i,j=0}^{N-1} C_k(i,j) * \hat{C}_k(i,j)}{\sum_{i,j=0}^{N-1} C_k(i,j)^2}$$

Structural Content

El contenido estructural de una imagen está definido para cada banda k como:

$$SC = \frac{\sum_{i,j=0}^{N-1} C_k(i,j)^2}{\sum_{i,j=0}^{N-1} \hat{C}_k(i,j)^2}$$

Spectral Measures

La transformada discreta de Fourier (DFT) de la imagen original y la imagen suavizada, denotadas como $\tau_k(u,v)$ y $\hat{\tau}_k(u,v)$ respectivamente están definidas como:

$$\tau_k(u,v) = \sum_{m,n=0}^{N-1} C_k(m,n) * e^{\left[-2\pi i m \frac{u}{N}\right]} * e^{\left[-2\pi i n \frac{v}{N}\right]},\ \text{k} = 1...\text{K}$$

$$\hat{\tau}_k(u,v) = \sum_{m,n=0}^{N-1} \hat{C}_k(m,n) * e^{\left[-2\pi i m \frac{u}{N}\right]} * e^{\left[-2\pi i n \frac{v}{N}\right]},\ \text{k} = 1...\text{K}$$

siendo K el número de bandas de la imagen. La fase y magnitud del espectro (de la transformada discreta de fourier) están definidos como:

$$\varphi(u,v) = arctan(\tau_k(u,v))$$
$$M(u,v) = |\tau_k(u,v)|$$

Con estos conceptos podemos definir las siguientes características para cada banda de la imagen:

- **Spectral Phase**

$$SP = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} |\varphi(u,v) - \hat{\varphi}(u,v)|^2$$

- **Spectral Magnitude**

$$SP = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left| M(u,v) - \hat{M}(u,v) \right|^2$$

- **Weighted Spectral Distance**

  Realiza una media ponderada entre la fase y la magnitud espectral:

  $WSD = \rho * SM + (1-\rho) * SP$

  En nuestro caso $\rho = 2.5 * 10^{-5}$.

Estas características también se pueden obtener para cada bloque de la imagen. Para ello consideramos que la imagen se divide en L bloques de tamaño bxb, y se calculan las características anteriores. De esta manera se pueden definir las siguientes características sobre el bloque l-ésimo para cada banda del bloque:

$$J^l{}_{\varphi} = \left(\sum_{u,v=0}^{b-1} \left(\varphi^l(u,v) - \hat{\varphi}^l(u,v) \right)^{\gamma} \right)^{\frac{1}{\gamma}}$$

$$J^l{}_{M} = \left(\sum_{u,v=0}^{b-1} \left(M^l(u,v) - \hat{M}^l(u,v) \right)^{\gamma} \right)^{\frac{1}{\gamma}}$$

$$J^l = \rho * {J^l}_M + (1-\rho) * {J^l}_\varphi$$

Para calcular estas características se utiliza $\gamma = 2$ y un tamaño de bloque de 32x32. Una vez calculadas estas medidas para cada bloque podemos obtener las siguientes características:

- ***Median Block Spectral Magnitude***

  $MBSM = mediana{J^l}_M,\ \mathrm{l} = 1...\mathrm{L}$

- ***Median Block Spectral Phase***

  $MBSM = mediana{J^l}_\varphi,\ \mathrm{l} = 1...\mathrm{L}$

- ***Median Block Weighted Spectral Distance***

  $MBWSM = medianaJ^l,\ \mathrm{l} = 1...\mathrm{L}$

Medidas basadas en el sistema visual humano (HVS)

Las imágenes pueden ser procesadas mediante filtros que simulan el HVS (*human visual system*). Uno de los modelos utilizados para ello es un filtro pasa banda con una función de transferencia en coordenadas polares:

$$H(\rho) = \begin{cases} 0.05e^{\rho^{0.554}} & \rho < 7 \\ e^{-9[|log_{10}\rho - log_{10}9|]^{2.3}} & \rho \geq 7 \end{cases}$$

donde $\rho = \sqrt{(u^2+v^2)}$. Se define el operador U como:

$$U\{C(i,j)\} = DCT^{-1}\left\{H\left(\sqrt{u^2+v^2}\right)\omega(u,v)\right\}$$

donde $\omega(u,v)$ denota la la transformada discreta del coseno bidimensional (DCT) de la imagen y DCT^{-1} es la DCT inversa bidimensional.

Algunas posibles medidas para cada banda de la imagen son:

- ***Normalized absolute error (HVS)***:

$$NAE = \frac{\sum_{i,j=0}^{N-1} \left| U\{C_k(i,j)\} - U\left\{\hat{C}_k(i,j)\right\}\right|}{\sum_{i,j=0}^{N-1} |U\{C_k(i,j)\}|}$$

- ***HVS based L2***:

$$L2 = \left\{ \frac{1}{N^2} \sum_{i,j=0}^{N-1} \left| U\{C_k(i,j)\} - U\left\{\hat{C}_k(i,j)\right\}\right|^2 \right\}^{\frac{1}{2}}$$

4.2.3 Estadísticas Wavelet

Se descompone cada banda de color de la imagen en tres sub-bandas utilizando QMF (*separable quadratic mirror filters*) y posteriormente se calcula la media de cada una de las tres sub-bandas dando como resultado un total de 9 características.

En la Figura 4.1 se observa los valores absolutos de los coeficientes de las sub-bandas para la imagen de un disco. Se puede ver arriba a la derecha la sub-banda vertical, abajo a la izquierda la sub-banda diagonal, y abajo a la izquierda la sub-banda diagonal. Esta imagen muestra una descomposición en tres niveles, en nuestro algoritmo sólo se usa el nivel uno.

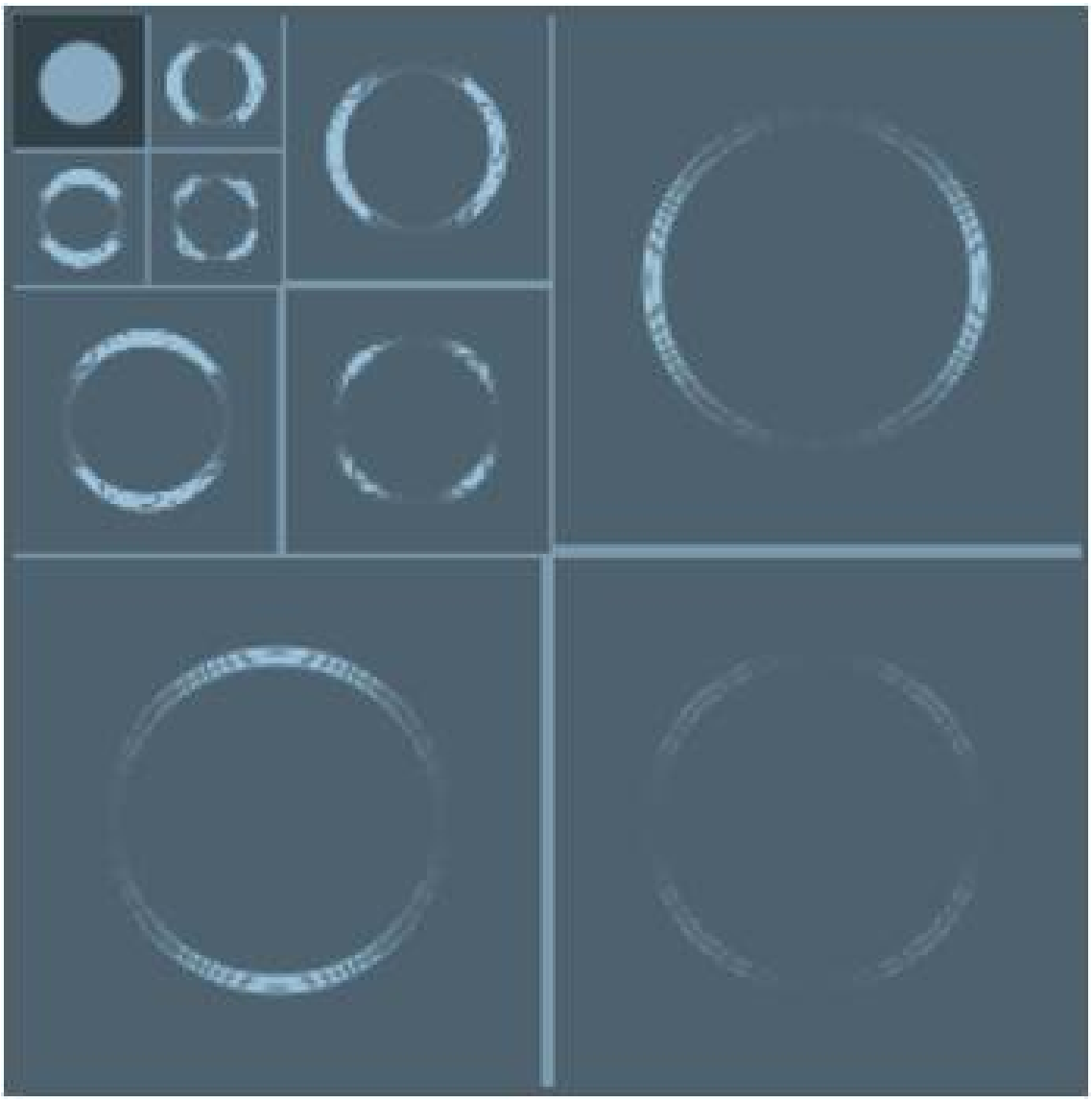


Figura 4.1: Ejemplo de subbandas Wavelet

Capítulo 5

Técnicas de Clasificación

En este apartado enumeraremos y explicaremos los clasificadores más importantes para la distinción de clases mediante características [BHvR03].

5.1 Clasificadores Basados en Distancias

Estos clasificadores se basan en el concepto de distancia entre los vectores de características [BHvR03].

Tenemos diferentes clasificadores basados en distancias como son:

- **Clasificador de distancia euclídea determinista a priori**: Es un clasificador determinístico, supervisado y a priori. Está basado en el cálculo de un prototipo (también denominado "centroide") para cada una de las K clases en las que se divide el universo de trabajo. Este prototipo puede verse como un representante "ejemplar" de cómo debería de ser un vector de características de esa clase. Así, ante un patrón desconocido se calcula la distancia euclídea del patrón que se desea clasificar a cada uno de los K prototipos. Un patrón desconocido X se clasificará como correspondiente a la clase cuyo prototipo esté a menor distancia según la distancia euclídea. Así, el clasificador euclídeo divide el espacio de características en regiones mediante "hiperplanos" equidistantes de los centroides.

- **Clasificador estadístico a priori**: En las situaciones en que los vectores de alguna clase presenten una dispersión significativa respecto a la media, o en aquéllas en las que no existe posible separación lineal entre las clases, puede ofrecer mejores resultados la sustitución de la distancia euclídea por la distancia de Mahalanobis. Esta medida, que tiene en cuenta la desviación típica de los vectores de características de los patrones de la muestra, puede proporcionar regiones de separación entre clases que sigan curvas cónicas. Por ello, este clasificador ofrece más garantías al tratar de separar patrones para los que no se encuentra una separación lineal. Además el clasificador estadístico proporciona probabilidades de pertenencia a las clases, por lo que se debe incluir en el grupo de los clasificadores no deterministas.

- **Clasificador con aprendizaje supervisado**: En este clasificador, el problema de la clasificación de un vector X en una de K clases se plantea como un problema de optimización. Más formalmente, para un conjunto de K clases ?1,?2,...?K la solución al problema de clasificar un vector X desconocido consiste en asociarlo a aquella clase cuya función discriminante fd i(X)) dé un resultado máximo.

5.2 Clasificadores Bayesianos

Este clasificador se fundamenta en la regla de Bayes del mínimo error [SNAPO07]. Un objeto, con unas características determinadas, pertenece a una clase si la probabilidad de pertenecer a ésta clase es mayor que la probabilidad de pertenecer a cualquier otra clase, como se muestra en la ecuación:

$$X \in \Omega_i si\ P(\omega)\,\rho\left(\frac{X}{\omega_i}\right) > P(\omega_j)\,\rho\left(\frac{X}{\omega_j}\right)$$

Donde ω es el espacio de características, que está dividido en regiones $\omega_i, i = 1, 2, ..., N$ donde N es el número de clases. $P(\omega_i)$ es la probabilidad a priori por la cual un objeto con características X, pertenece a la clase ω_i y $\rho\left(\frac{X}{\omega_i}\right)$ es la función de probabilidad condicional de la clase ω_i para X. En la práctica, las funciones de probabilidad no se conocen y por lo tanto se deben estimar. Para estimarlas, primero se asume la forma de la función de probabilidad, y luego se hallan sus parámetros a partir del conjunto de entrenamiento.

5.3 Clasificadores de Redes Neuronales

Las redes neuronales son una técnica de aproximación paramétrica útil para construir modelos de densidad.

Esta red presenta una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa de neuronas ocultas internas. Cada neurona (menos en la capa de entrada) recibe entrada de todas las neuronas de la capa previa y genera salida hacia todas las neuronas de la capa siguiente (salvo las de salida). No hay conexiones hacia atrás (*feedback*) ni laterales o autorrecurrentes. El funcionamiento de la red consiste en un aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo, empleando un ciclo propagación-adaptación de dos fases:

Primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener y se calcula un valor del error para cada neurona de salida. Estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada, es decir, que el error disminuya.

A diferencia de la regla delta en el caso del perceptrón, esta técnica requiere el uso de neuronas cuya función de activación sea continua y por tanto diferenciable. Generalmente la función será de tipo sigmoidal.

5.4 Algoritmos de Agrupamiento (*Clustering*)

En ocasiones no existe la figura del maestro que determina los patrones que pertenecen a una clase o a otra. En estos casos los algoritmos de agrupamiento o clustering permiten realizar esta tarea de manera automática. Estos algoritmos también se conocen como algoritmos de clasificación autoorganizados [BHvR03].

Los algoritmos de agrupación de clases se suelen utilizar cuando no existe conocimiento a priori de las clases en que se pueden distribuir los objetos, cuando las clases no son interpretables por un humano, o cuando el número de clases es muy elevado para un procesado no automático.

Algunos ejemplos de algoritmo son:

- **Algoritmo de distancias encadenadas**: El algoritmo de las distancias encadenadas construye una cadena partiendo de un patrón al azar y encadenando cada vez el patrón que esté más cerca del extremo de dicha cadena. El algoritmo crea automáticamente nuevas clases cuando la distancia entre dos patrones consecutivos supera cierto umbral. Para fijar la sensibilidad en la determinación de las clases, este algoritmo necesita del ajuste previo de dicho umbral.

- **Algoritmo MaxMin**: El algoritmo MaxMin elige paulatinamente representantes de entre los patrones de muestra, determinando la distancia a la que se encuentran el resto de patrones de los mismos. Si para algún patrón esta distancia supera cierto umbral se crea una nueva clase con ese patrón como representante. El proceso se repite hasta que no se producen cambios.

  Es un algoritmo tiene la ventaja de que sólo precisa la determinación de un umbral para la autoorganización de las clases. Además este valor puede interpretarse como el radio de las clases entorno al representante de cada una.

- **Algoritmo de las K-medias**: El algoritmo k-medias permite determinar la posición de k centroides que distribuyan de manera equitativa un conjunto de patrones. Debe notarse que, a diferencia de los algoritmos anteriores, este algoritmo tiene la particularidad de necesitar conocer a priori el número k de clases existentes.

5.5 *Support Vector Machines* (SVM)

Los algoritmos SVM pertenecen a la familia de los clasificadores lineales. En estos clasificadores tenemos la característica de que, a priori, conocemos las clases a las que pertenecen nuestros individuos, no se trata de una agrupación por similitudes, sino que tenemos las clases bien definidas [BHvR03].

Dado un conjunto de ejemplos de entrenamiento (muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase, dependiendo de la proximidad a cada una.

Más formalmente, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación más correcta.

Matemáticamente:

Partimos de un conjunto de datos de entrenamiento $\{x_i, y_i\}$ con:

$$i = 1, ..., l, y_i \in \{-1, 1\} \text{ y } x_i \in R^d$$

Entonces existe un hiperplano, como el de la figura que separa los datos de etiquetas positivas y negativas, tales que:

$$x_i\omega + b \geq 1 - \xi_i para y_i = 1 \; x_i\omega + b \leq 1 + \xi_i para y_i = -1 \; _i \geq \forall i$$

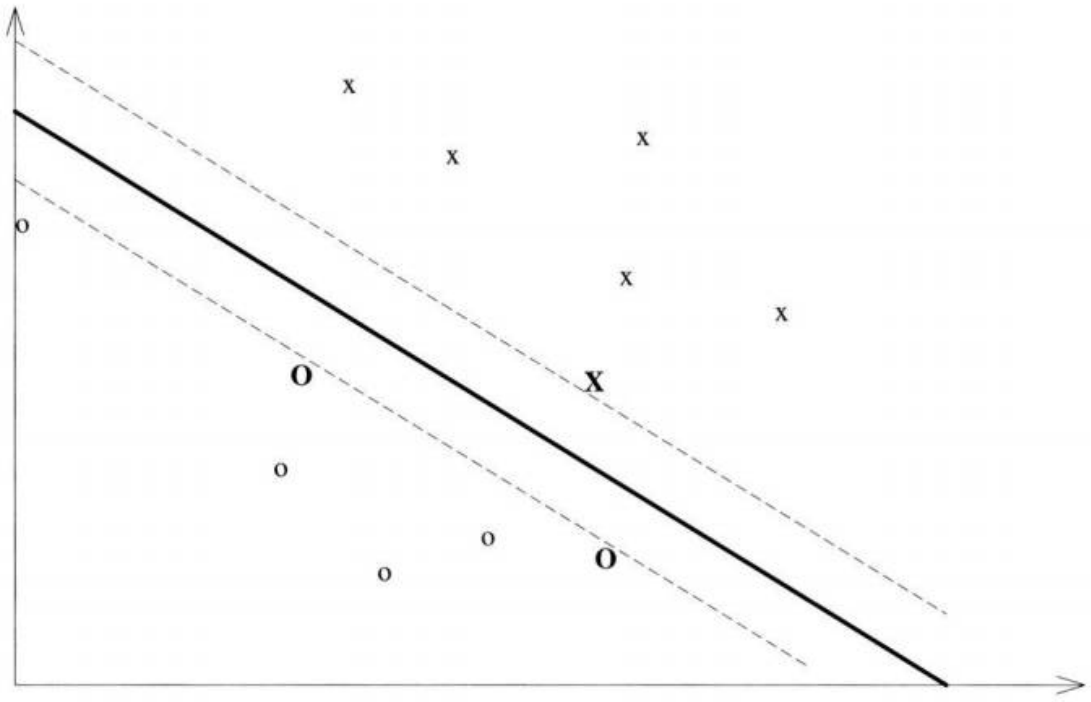


Figura 5.1: Representación gráfica de máquinas de soporte vectorial

Donde ω es la normal al hiperplano y $_i$ son las variables introducidas por los errores de clasificación en calidad de violaciones del hiperplano, de manera que $\sum \xi_i$ será la cota del error de clasifcación. Una manera directa de añadir el coste a la función objetivo es minimizar $\frac{\|\omega^2\|}{2} + C\sum \xi_i$, siendo C la constante elegida correspondiente al inverso del valor de la penalización de los errores. Así, se tiene un caso de optimización convexa cuyo problema de optimización cuadrática es el número de vectores soporte.

En la mayoría de los casos, el espacio de entrada no es lineal, y por tanto es necesario hacer su transformación a un espacio euclídeo H (ver Figura 5.2).

Por consiguiente, el algoritmo de entrenamiento solo depende de los datos de entrada, a través de los productos de la forma $\phi(x_i) : \phi(x_j)$. Existirá entonces una función llamada "kernel" tal que se cumple que:

$$K(x_i, x_j) = \phi(x_i) : \phi(x_j)$$

De los "kernel" usados más comunes tenemos:

- Polinomial (homogeneo) $K(x_i, x_j) = (x_i \cdot x_j)^d$
- Polinomial(heterogéneo) $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$
- RBF (Radial Basis Function) $K(x_i, x_j) = exp\left(-\gamma \|x_i - x_j\|^2\right)$
- Tangente hiperbólica $K(x_i, x_j) = tanh(kx_i \cdot x_j + c)$ para algún $k > 0$ y $c < 0$

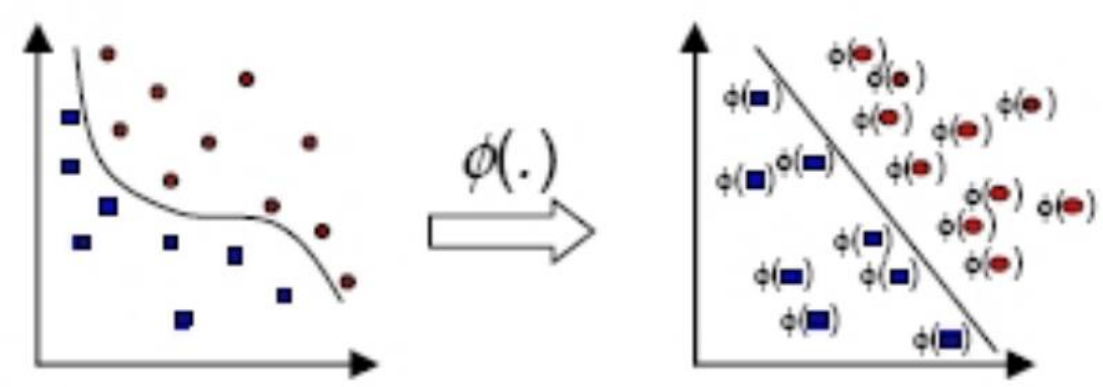


Figura 5.2: Transformación a espacio euclídeo

En la Tabla 5.1 se presenta una comparativa de SVM con los algoritmos basados en redes neuronales.

Tabla 5.1: Conjunto de Image Quality Metrics

Algoritmos redes neuronales	**SVM**
Capas ocultas transforman a espacios de cualquier dimensión	Kernels transforman a espacios de dimensión muy superior
El espacio de búsqueda tiene múltiples mínimos locales	
	El espacio de búsqueda tiene sólo un mínimo global
El entrenamiento es costoso	El entrenamiento es muy eficiente
La clasificación es muy eficiente	La clasificación es muy eficiente
Se diseña el número de capas ocultas y nodos	Se diseña la función “kernel” y el parámetro de coste C
Muy buen funcionamiento en problemas típicos	Muy buen funcionamiento para problemas típicos

Capítulo 6

Implementación de los Algoritmos

En la implementación se ha utilizado el lenguaje Python y posteriormente se paso en su mayoría a C para conseguir las características rápidamente ya que python no daba la suficiente velocidad, sobre todo para imágenes grandes.

Todo este proceso se intento hacer de manera óptima aprovechando al máximo los cálculos, por ejemplo, en la parte del algoritmo de obtención de características de ruido, el filtro gaussiano utilizado está totalmente en C, pero tiene varias modificaciones para acelerar los cálculos de las correlaciones y aprovechar lo máximo posible los diferentes recorridos que se hacen de la imagen tanto sobre filas como columnas.

También se cuenta con ayuda de las siguientes librerías:

- **Numpy**: librería python para la realización de cálculos matemáticos. Proporciona una potente clase para manejar arrays multidimensionales compatible con C, además de otras funciones muy útiles para el proyecto como la transformada discreta de Fourier.
- **Scipy**: extensión a numpy que proporciona funciones de diferentes campos científicos. Contiene funciones útiles como el centro de masa de un array multidimensional o la transformada discreta del coseno.
- **Pywavelets**: librería python usada para descomponer la imagen en sub-bandas necesarias para las características wavelet.
- **Python Imaging Library**: librería que facilita el manejo de imágenes en python. Admite diversos formatos de imagen e incluye funciones para el procesamiento de imágenes.

6.1 Mejora de Rendimiento: Módulos C y Paralelización

Durante la implementación del código para la extracción de las características de las imágenes nos dimos cuenta de que debido a la gran cantidad de datos que hay que manipular para la obtención de dichas características (para cada característica hay que procesar todos y cada uno de los píxeles, por ejemplo, en una foto de 5 megapíxeles tenemos que hacer cálculos con 5 millones de píxeles) Python se demoraba en exceso. Esto es debido a la lentitud del lenguaje a la hora de realizar cálculos en los que hay bucles con muchas iteraciones.

La solución fue la utilización del lenguaje C para la implementación de aquellas partes del código que tuviesen una gran complejidad de cómputo, ya que este lenguaje al ser

código compilado y no utilizar un intérprete como Python consigue un rendimiento muy superior en cuanto a velocidad de cálculo.

A todo esto y para incrementar aún más el rendimiento se paralelizaron la obtención de características, aprovechando los sistemas multinúcleo actuales.

6.1.1 Módulos C

La combinación de código C y código Python se realiza en cuatro pasos:

1. Creo el código Python desde el cual voy a llamar a la función o funciones que implemente en C.
2. Mando los datos al módulo C haciendo una llamada a las funciones de dicho módulo.
3. Dentro del módulo tendré el algoritmo para procesar los datos.
4. Una vez que he procesado los datos y generado una solución, tengo que mandar el resultado de vuelta a Python.

Implementación del Módulo

En la Figura 6.1 se presenta un ejemplo de un módulo implementado en C.

Lo que hace este código es obtener lo que le enviamos desde Python en la función funciones_structuralContent a través del parámetro args, que es una tupla de Python con todos aquellos parámetros que queremos mandar a la función. Para separar los datos contenidos en esta tupla se utiliza la función PyArg_ParseTuple(), que tiene como entrada a args, una cadena de texto con los tipos de las variables y finalmente un puntero a las variables de C donde queremos almacenar los parámetros de entrada. En este caso, al tratarse de dos objetos de tipo ndarray de la librería numpy ponemos como tipo la cadena "OO" ('O' es para objetos, 'i' para enteros, 'd' para doublés, etc) y los almacenamos en variables de tipo PyObject para posteriormente hacer un casting a PyArrayObject. Una vez hecho esto ya puedo utilizar las funciones de la clase ndarray con los datos de entrada. Todo esto explica la realización del paso 2.

Posteriormente tenemos el código C que ejecuta el algoritmo con los datos (paso 3).

Finalmente tengo que mandar el resultado a Python (paso4). Para ello utilizo la función Py_BuildValue() que toma como parámetros una cadena de texto con los tipos de los datos a devolver y las variables que contienen dichos datos, análogo a como se hacía con PyArg_ParseTuple pero al revés, esta vez lo que devuelve es un objeto (PyObject) de tipo tupla. En el código de ejemplo se devuelve una tupla con tres valores de tipo double.

Dentro del módulo también hay que añadir la tabla de métodos, en la cual incluiremos todos los método que queremos que se ejecuten desde python de la siguiente manera:

"Nombre de función que usaremos en Python", Nombre de la función en C, METH_VARARGS, "Comentario para explicar el uso de la función"

Finalmente, tenemos que crear una función con nombre initNombreDelModulo(), que se encarga de inicializar el módulo con la tabla de métodos que hemos creado antes.

Compilación del Módulo y su Uso en Python

Para poder utilizar el módulo antes debemos compilarlo. Para ello creamos un archivo setup.py como el mostrado en la Figura 6.2:

Dentro de este archivo tenemos que introducir el nombre del módulo ("funciones") y el nombre del archivo a compilar ("funcionesC.c").

```c
#include <Python.h>
#include <numpy/arrayobject.h>
#include <math.h>

static PyObject *funcionesC_structuralContent(PyObject *self,PyObject *args){
    PyObject *imOObject,*imSObject;
    PyArrayObject *imagenO,*imagenS;
    PyArg_ParseTuple(args,"OO", &imOObject, &imSObject);
    imagenO = (PyArrayObject *)imOObject;
    imagenS = (PyArrayObject *)imSObject;
    int alto = PyArray_DIMS(imagenO)[1];
    int ancho = PyArray_DIMS(imagenO)[2];
    int i,j;
    double sumRO=0, sumGO=0, sumBO=0, sumRS=0, sumGS=0, sumBS=0;
    for(i=0;i<alto;i++){
        for(j=0;j<ancho;j++){
            sumRO = sumRO + pow((*(unsigned char*)PyArray_GETPTR3(imagenO,0,i,j)),2);
            sumGO = sumGO + pow((*(unsigned char*)PyArray_GETPTR3(imagenO,1,i,j)),2);
            sumBO = sumBO + pow((*(unsigned char*)PyArray_GETPTR3(imagenO,2,i,j)),2);
            sumRS = sumRS + pow((*(unsigned char*)PyArray_GETPTR3(imagenS,0,i,j)),2);
            sumGS = sumGS + pow((*(unsigned char*)PyArray_GETPTR3(imagenS,1,i,j)),2);
            sumBS = sumBS + pow((*(unsigned char*)PyArray_GETPTR3(imagenS,2,i,j)),2);
        }
    }
    double scR = sumRO / sumRS;
    double scG = sumGO / sumGS;
    double scB = sumBO / sumBS;

    return Py_BuildValue("ddd",scR,scG,scB);
}

static PyMethodDef funcionesCMethods[] = {
    {"structuralContent", funcionesC_structuralContent, METH_VARARGS, "Structural Content"},
    {NULL, NULL, 0, NULL}
};

void initfuncionesC(void){
    (void) Py_InitModule("funcionesC", funcionesCMethods);
}
```

Figura 6.1: Código necesario en C

```python
from distutils.core import setup, Extension

module =Extension('funcionesC', sources = ['funcionesC.c'])
setup(name = 'Prueba C', version = '1.0', ext_modules = [module])
```

Figura 6.2: Código necesario en setup.py

Ahora desde la consola, con el código C en el mismo directorio escrbimos:

python setup.py build

Eso creará una carpeta "build" en nuestro directorio, y dentro una carpeta llamada "lib.SO-VerPython" donde SO es nuestro SO y VerPython es la versión de Python para la que esta compilado nuestro módulo. En esta carpeta encontraremos el archivo .pyd. Ahora

solo hay que importar el módulo desde nuestro código python.

Para importar el módulo desde python solo hay que hacer un import. En nuestro caso "import funcionesC" y a partir de ese momento ya se pueden utilizar todas las funciones implementadas en él. Esto se correspondería con el paso 1 anteriormente mencionado.

Resultados Obtenidos

En lo relacionado con el proyecto, se creó un módulo en C llamado funciones con el fin de mejorar la velociadad de ejecución de la extracción de las características necesarias para el entrenamiento del SVM. Este módulo consta de las siguientes funciones:

- **gauss(imagenEntrada, nucleo, imagenResultado)**: esta función ejecuta el bucle que aplica el núcleo a la imagen de entrada.
- **czekonowskyDistance(imagenO, imagenS)**: obtiene la distancia Czekonowsky existente entre la imagen original y la imagen suavizada (imagen tras aplicar el filtro gaussiano).
- **minkowskyMetric(imagenO, imagenS, gamma)**: obtiene el mean absolute error (para gamma=1) y el mean square error (para gamma=2) entre la imagen original y la imagen suavizada.
- **structuralContent(imagenO, imagenS)**: obtiene el contenido estructural entre la imagen original y la imagen suavizada.
- **normalizedCrossCorrelation(imagenO, imagenS)**: obtiene la correlacion cruzada normalizada de la imagen original y la imagen suavizada.
- **laplacianMeanSquareError(imagenO, imagenS)**: obtiene el laplacian mean square error entre la imagen original y la imagen suavizada.
- **neighborDistribution(imagenO, vectorSalida)**: calcula la distribución de vecindad de la imagen original en vectorSalida, utilizado para calcular el centro de masa de la distribución de vecindad.
- **bucle1HVS() y bucle2HVS()**: fragmentos de código utilizados para calcular las medidas HVS de manera mas rápida.
- **spectralPhase(dftO, dftS) y spectralMagnitude(dftO, dftS)**: calcula la spectral phase y la spectral magnitude a partir de la transformada discrete de fourier de la imagen original y la imagen suavizada.
- **siguienteBloque(posX, posY, N, M, tamañoBloque)**: utilizado para las median block features, se encarga de devolver las coordenadas del siguiente bloque a procesar teniendo en cuenta la posición actual, el tamaño de la foto y el tamaño de bloque que queramos.
- **bucleMBF(dftO, dftS, gamma, lambd)**: contiene código para agilizar la obtención de las median block features.
- **pearson(imagen1, imagen2)**: calcula el coeficiente de correlación de Pearson para una determinada banda de dos imágenes.

Algunos de los resultados obtenidos para una foto de 8 megapíxeles se presentan en la Figura 6.3.

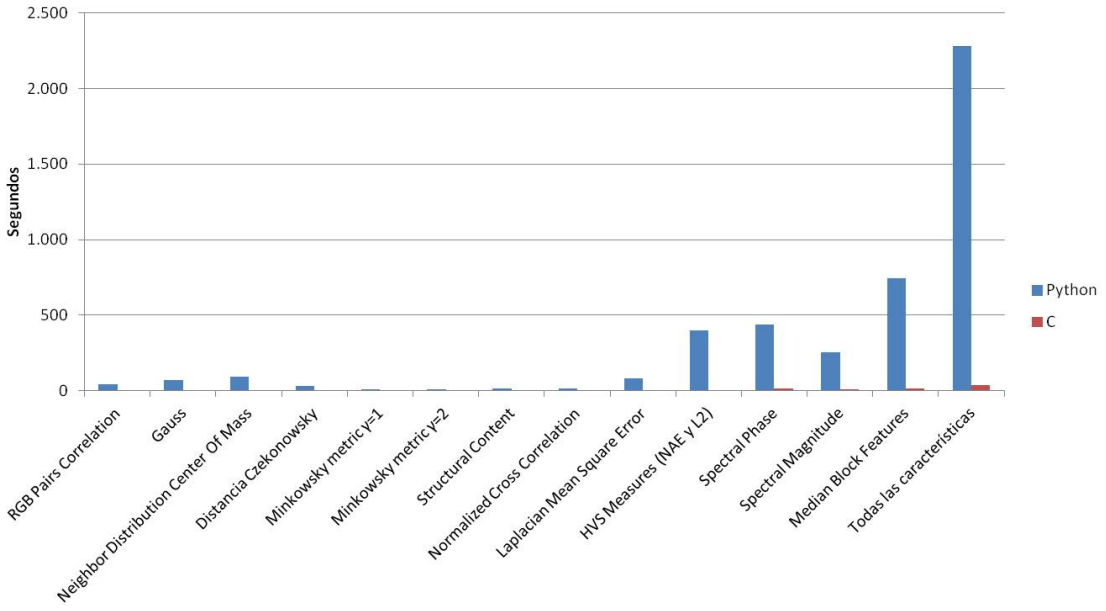


Figura 6.3: Comparativa de ejecución de módulos desarrollados en C y Python

6.1.2 Paralelización de Características

Para mejorar aún más el rendimiento de la extracción de características, el siguiente paso que se dio fue la paralelización de estas.

En primera instancia se intento aprovechar la potencia de las GPUs actuales y hacer uso de OpenCL para la extracción de características. Con esta librería se podrían aprovechar los numerosos núcleos de las tarjetas gráficas para procesamiento matemático. Esta idea tenía su parte negativa y era la dificultad de implementación de las diferentes funciones para que se pudieran aprovechar de manera óptima esta potencia, además de ser necesario un equipo con una GPU compatible, moderna y potente para que esta ventaja se obtuviera.

Posteriormente se estudio la posibilidad natural de paralelización con threads. Python en este sentido tenía un gran problema y era que, aún teniendo una api para su manejo, no se podían ejecutar varios threads en paralelo.

La ejecución de los threads en Python está controlada por el GIL (*Global Interpreter Lock*) de forma que sólo un thread puede ejecutarse a la vez, independientemente del número de procesadores con el que cuente la máquina. Esto posibilita que el escribir extensiones en C para Python sea mucho más sencillo, pero tiene la desventaja de limitar mucho el rendimiento, por lo que a pesar de todo, en Python, en ocasiones nos puede interesar más utilizar procesos que threads, que no sufren de esta limitación [PAR].

Por este motivo no se pudo paralelizar la aplicación a nivel de threads y se tuvo que hacer a nivel de proceso. En esta parelización se tuvo que hacer partes con lo denominado "exclusión mutua" en la parte final de cada proceso, donde se iban recogiendo los datos comunes y necesitábamos que solo un proceso exclusivamente realizara esa parte. Toda esta gestión de los threads fue posible gracias a la ayuda del módulo processing incluido por defecto en python.

En cuando al número de procesos a lanzar, depende del número de núcleos que tenga el computador en cuestión, aunque debido a la cantidad de memoria que usaba cada proceso (cuyo uso se incrementaba masivamente según crecía el tamaño de la fotografía)

en algunos caso convenía limitarlo a un número de procesos menor de este valor para obtener un rendimiento óptimo.

Los resultados obtenidos (en segundos) para la extracción de las características de distinción entre modelos de cámaras, en un ordenador con procesador Intel i7-720 con 4 núcleos (8 hilos de ejecución), una velocidad de entre 1.6Ghz a 2.8Ghz dependiendo del número de hilos que se estén ejecutando en ese momento y 4GB de ram, se presentan en la Figura 6.4.

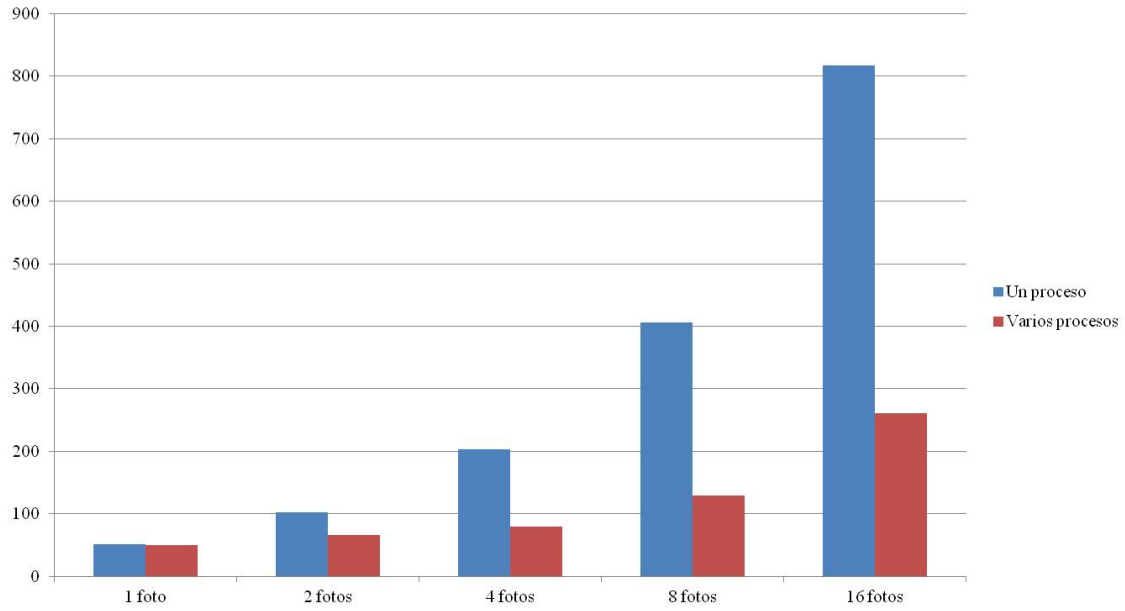


Figura 6.4: Comparativa de rendimiento en la extracción de características

Como se puede observar en la Figura 6.4 el aumento de rendimiento es igual con 8 fotos y con 16, ya que el procesador puede ejecutar hasta 8 procesos simultáneamente.

Capítulo 7

Experimentos y Resultados

En este capítulo hablaremos sobre los diferentes experimentos realizados y los resultados obtenidos. Primero se realizaron experimentos con grupos de imágenes de escáneres, móviles y generadas por computador. Seguidamente nos centraremos en experimentos sobre modelos de móvil concretos con el algoritmo de 61 características extraídas.

7.1 Experimentos y Resultados con Imágenes de Escáner, Móviles y Generadas por Computador

En este último apartado hablaremos sobre como se hicieron los diferentes experimentos, con diferentes conjuntos de imágenes y dentro de estas tomándolas de diferentes maneras: originales, redimensionadas o recortadas. Utilizando también diferentes conjuntos de características y mostrando los resultados y conclusiones obtenidas con todos estos experimentos.

Los diferentes conjuntos de imágenes serán principalmente 3:

- Imágenes tomadas desde teléfonos móviles
- Imágenes obtenidas con un escáner
- Imágenes generadas digitalmente por un computador

En cuanto a las características utilizaremos 2 grandes conjuntos:

- Características centradas en la distinción del modelo de cámara de fotos
- Características basadas en correlaciones de ruido sobre filas y columnas, orientadas a la distinción entre escáneres y cámaras de fotos

7.1.1 Obtención de los Diferentes Conjuntos de Imágenes

Para obtener resultados reales, lo primero que debemos tener son grandes conjuntos de imágenes de cada tipo. Inicialmente se tuvieron conjuntos de imágenes pequeños, unos 30 elementos por grupo y esto distorsionaba en gran medida los resultados de las diferentes pruebas, teniendo en cuenta que se quieren dar resultados que representen a grupos completos.

Hay que tener presente que debemos usar fotos que conozcamos seguro que pertenecen a un determinado grupo, en caso contrario descartarlas. Si no prestamos especial atención a esto, aún siendo pocas las fotos en las que pueda ocurrir esto, nos producirá ruido,

dándonos peores porcentajes de acierto, tanto en la fase de entrenamiento como de testeo. Esto se acentuará más si los conjuntos de imágenes son más pequeños.

En nuestros conjuntos de imágenes, en las fotos tomadas desde teléfonos móviles, no hay ningún inconveniente en obtener cualquier número de fotos de diferentes móviles. En los otros grupos esta tarea es más complicada.

Se decidió usar para estos grupos, imágenes descargadas desde Flickr. En las imágenes generadas por computador, con este método, no había excesivos problemas en conseguir imágenes en gran cantidad, por lo que el conjunto más crítico que se tiene es el de imágenes escaneadas.

Para este grupo se descargaron imágenes de grupos o imágenes con tags tipo "scanned images" en los que se asociaban imágenes de este tipo. Con este método se descargaron un total de 20532 imágenes posibles de escáner, unos 16,2 GB. El programa utilizado para esto fue "Saleen Flickr Downloader" el cual permite descargar imágenes de grupos o con diferentes "tags" de forma masiva.

Después de esto se aplicaron varios filtros para obtener un conjunto final de imágenes correcto.

Los diferentes filtros fueron:

1. Imágenes originales, las imágenes descargadas que son originales (y no redimensionas) terminan en "_O". También, en este filtro se conservaron solo las imágenes con un tamaño mínimo de lado de 1024 píxeles (de ancho o de alto). Con este filtrado nos quedamos con 8426 imágenes.

2. Imágenes con "tag" modelo de cámara rellenado. Descartamos imágenes que no sabemos qué dispositivo la produjo. Nos quedamos con 1229 imágenes.

3. Con tag "orientación" no vacio y que este tenga el valor "orientación normal". Esto en imágenes generadas por escáner es fundamental ya que las fotos giradas 90 grados (a izquierda o derecha) cambian totalmente las características ya que se invierten filas y columnas. Nos quedan 571 imágenes.

4. Imágenes con tag "modelo de cámara" con valor a un escáner normal conocido, no se incluyen escáneres de negativos o similares. Con este filtro nos quedamos con 287.

5. Imágenes de menos de 8 Mpx. Hacemos esto para descartar imágenes gigantes. Nos quedamos con 271 imágenes.

6. Imágenes en color de 24 bits, así descartamos las pocas imágenes que quedan en blanco y negro. Nos quedamos finalmente con 260 imágenes.

Debido a la gran variedad de tamaños que tenían nuestras imágenes, por tener un tamaño más uniforme, se cortaron las imágenes en trozos de 1024x768, de la siguiente manera:

Si nuestra imagen es más grande de 1024x768 cortamos la imagen a esa resolución desde la esquina superior izquierda, si podemos recortar otra de ese mismo tamaño por debajo de la anterior o hacia la derecha lo hacemos y así sucesivamente. No cogemos ningún trozo que sea inferior a 1024 de ancho y 768 de ancho.

Con este último paso volvemos a recuperar varias imágenes. Por último, para terminar de sacar un conjunto aceptable de imágenes cogemos las imágenes que no estén muy comprimidas y que por tanto tengan más detalles y sean más encasillables en un grupo. Con esto fijamos el valor mínimo del tamaño la imagen a 100KB, que nos vuelve a dar

un total de unas 260 imágenes, el cual será el número de imágenes que tomemos de cada grupo.

Para las imágenes generadas por teléfono móvil se escogieron directamente 260 imágenes de tamaño mayor o igual que 1024x768 y menores de 2 Mpx (1600x1200), de diferentes móviles de diferentes marca y con un tamaño mínimo de 100 KB.

En las imágenes generadas por ordenador se uso la misma forma de obtenerla que las de móviles (descargadas de flickr) con los mismos tamaños límite y el mismo tamaño mínimo. En estas imágenes se han descartado las imágenes con el "tag" modelo de cámara con un valor de una cámara o escáner.

7.1.2 Experimentos con Algoritmo para Distinción de Imágenes por Correlaciones de Ruido sobre Filas y Columnas

Este algoritmo está orientado a la distinción entre imágenes de cámaras digitales e imágenes generadas por un escáner basándose en la variación de ruido sobre las diferentes filas y columnas. Probaremos combinaciones de los 3 grupos de imágenes, con diferentes resoluciones, obtenidas estás, por redimensionado o recorte de las imágenes originales. El número de imágenes siempre será de 260 en cada grupo salvo algún caso excepcional.

Experimentos de Imágenes de Escáneres vs Móviles

Los resultados (porcentaje de aciertos) de las pruebas imágenes Vs móviles se recogen en la Tabla 7.1.

Tabla 7.1: Experimento de imágenes de escáneres vs imágenes de móviles (I)

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
1	Imágenes originales de escáner	Imágenes originales de móviles	1024x768	$\geq$ 1024x768 y $<$ de 2Mpx (1600x1200)	260	95.7854 %
2	Imágenes recortadas de escáner	Imágenes originales de móviles	640x480	$\geq$ 1024x768 y $<$ de 2Mpx (1600x1200)	260	96.16 %
3	Imágenes recortadas de escáner	Imágenes originales de móviles	400x300	$\geq$ 1024x768 y $<$ de 2Mpx (1600x1200)	260	96.737 %

La prueba 1 compara los conjuntos de imágenes originales. En el resto de pruebas de la Tabla 7.1 mantenemos las imágenes de móviles y vamos haciendo más pequeñas mediante sucesivos recortes las imágenes obtenidas mediante escáner.

Como podemos observar los porcentajes de aciertos son muy altos, ya que el algoritmo está orientado a distinguir imágenes de estos grupos. Cuando reducimos la resolución de

las escaneadas vemos que debido a la incremental diferencia de resoluciones la distinción se realiza ligeramente mejor, aunque no en gran medida.

En el experimento mostrado en la Tabla 7.2 comenzamos con conjuntos de imágenes diferentes a los de referencia. Tenemos 100 imágenes solo de cada grupo partiendo de imágenes a 1.3 Mpx en ambos casos. El porcentaje de acierto con ambas originales (prueba 4) es bastante inferior a la prueba anterior. Realizamos las mismas operaciones que en el experimento anterior. Vemos que al tener un conjunto de imágenes bastante inferior los resultados son más inestables y tienen cambios mucho más bruscos, al realizar los mismos cambios que en el caso anterior. En la Tabla 7.1 solo variaba un 1 %, en este caso varía casi un 8 % desde el principio de cada prueba.

Tabla 7.2: Experimento de imágenes de escáneres vs imágenes de móviles (II)

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
4	Imágenes originales de escáner	Imágenes originales de móviles	1.3 Mpx	1.3 Mpx	100	89.756 %
5	Imágenes recortadas de escáner	Imágenes originales de móviles	1024x768	1.3 Mpx	100	93.1707 %
6	Imágenes recortadas de escáner	Imágenes originales de móviles	640x480	1.3 Mpx	100	96.0976 %
7	Imágenes recortadas de escáner	Imágenes originales de móviles	400x300	1.3 Mpx	100	97.0732 %

En la Tabla 7.3 vemos los resultados de recortar ambos conjuntos.

En la prueba 8 tomamos la imágenes originales de escáneres y las comparamos con las de móviles recortadas a 1024x768. Vemos que esto reduce mucho el porcentaje de aciertos. En las pruebas 9 y 10 reducimos recortando ambos conjuntos, desde la prueba 8.

Vemos que, cuanto más pequeñas son las imágenes que tenemos en los conjuntos, peor las distingue, al tener menos detalles y datos en estas.

Probamos ahora reduciendo las imágenes de los móviles contra las imágenes de escáneres recortadas. La prueba 11 difiere de la 8 en que las imágenes de móvil han sido redimensionadas en vez de recortadas. Esto nos da un resultado esperable y es que la imágenes de móviles conservan mejor sus características (y por tanto se distinguen mejor) cuando son redimensionadas a cuando son troceadas. Vemos también que a 640x480 incluso sube el porcentaje de aciertos y que si bajamos más la resolución el porcentaje de aciertos cae bruscamente (ver Tabla 7.4).

Por último, hacemos un experimento reduciendo ambos grupos de imágenes. Vemos en la Tabla 7.5 que de esta forma los porcentajes se mantienen por encima del 92.5 %, incluso suben ligeramente en el caso de imágenes reducidas a 400x300.

Tabla 7.3: Experimento de imágenes de escáneres vs imágenes de móviles (III)

Prueba	Grupo 1	Grupo 2	Resolución 1	Resolución 2	Imágenes por grupo	% Aciertos
8	Imágenes originales de escáner	Imágenes recortadas de móviles	1024x768	1024x768	260	90.0763 %
9	Imágenes recortadas de escáner	Imágenes recortadas de móviles	640x480	640x480	260	88.7405 %
10	Imágenes recortadas de escáner	Imágenes recortadas de móviles	400x300	400x300	260	85.3053 %

Tabla 7.4: Experimento de imágenes de escáneres vs imágenes de móviles (IV)

Prueba	Grupo 1	Grupo 2	Resolución 1	Resolución 2	Imágenes por grupo	% Aciertos
11	Imágenes originales de escáner	Imágenes reducidas de móviles	1024x768	1024x768	260	92.5573 %
12	Imágenes recortadas de escáner	Imágenes reducidas de móviles	640x480	640x480	260	92.9524 %
13	Imágenes recortadas de escáner	Imágenes reducidas de móviles	400x300	400x300	260	87.7863 %

Experimentos de Imágenes Generadas por Computador vs Móviles

El algoritmo para la distinción de imágenes por correlaciones de ruido no fue pensado para imágenes generadas por computador, sin embargo, teniendo en cuenta que estas imágenes deberían estar exentas de ruido, deberían poder distinguirse por este patrón.

En la primera prueba comparamos las imágenes originales de ambos grupos. Como se observa en la Tabla 7.6 el porcentaje de aciertos el bastante inferior al de las pruebas del apartado anterior bajando casi un 13 %.

En las pruebas 2, 3 y 4 redimensionamos ambos conjuntos de imágenes. Como es de esperar el porcentaje de aciertos se reduce, pero encontramos un límite justo por encima de 77.5 %.

Repetimos el mismo proceso pero recortando las imágenes en vez de redimensionarlas. Con esto, vemos en la Tabla 7.7 que inicialmente el porcentaje de aciertos se reduce,

Tabla 7.5: Experimento de imágenes de escáneres vs imágenes de móviles (V)

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
14	Imágenes reducidas de escáner	Imágenes reducidas de móviles	640x480	640x480	260	92.5573 %
15	Imágenes reducidas de escáner	Imágenes reducidas de móviles	400x300	400x300	260	92.9524 %

Tabla 7.6: Experimento imágenes generadas por computador Vs móviles (I)

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
1	Imágenes originales generadas por computador	Imágenes originales de móviles	$\geq$ 1024x768 y $<$ de 2Mpx (1600x1200)	$\geq$ 1024x768 y $<$ de 2Mpx (1600x1200)	260	82.9317 %
2	Imágenes reducidas generadas por computador	Imágenes reducidas de móviles	1024x768	1024x768	260	80.0797 %
3	Imágenes reducidas generadas por computador	Imágenes reducidas de móviles	640x480	640x480	260	77.8218 %
4	Imágenes reducidas generadas por computador	Imágenes reducidas de móviles	400x300	400x300	260	77.6031 %

pero después el descenso en bastante más suavizado superando con resolución 640x480 al experimento anterior y manteniéndose prácticamente constante entre 78 % y 79 %.

Tabla 7.7: Experimento imágenes generadas por computador Vs móviles (II)

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
5	Imágenes recortadas generadas por computador	Imágenes recortadas de móviles	1024x768	1024x768	260	79.9603 %
6	Imágenes recortadas generadas por computador	Imágenes recortadas de móviles	640x480	640x480	260	79.7619 %
7	Imágenes recortadas generadas por computador	Imágenes recortadas de móviles	400x300	400x300	260	78.5571 %

Experimentos de Imágenes Generadas por Computador vs Escáneres

Intuitivamente una distinción por características de ruido debería funcionar mejor con estos dos grupos de imágenes, ya que las imágenes escaneadas tienen más ruido que la tomadas desde un móvil y la generadas por ordenador no deberían tener nada de ruido, al considerarse perfectas en ese aspecto.

En la primera prueba probamos los conjuntos originales. Vemos que el porcentaje de aciertos es inferior al anterior caso, contradiciendo lo que nos podría parecer más lógico.

En las siguientes pruebas redimensionamos las imágenes. En este caso, como se observa en la Tabla 7.8 ocurre que a reduciendo ambas a 1024x768 el porcentaje de aciertos se reduce, pero si reducimos más, a 640x480 se iguala con el inicial y a 400x300 supera por casi un 1 % a los conjuntos originales.

Con la siguiente prueba, recortando las imágenes de ambos conjuntos vemos en la Tabla 7.9 que inicialmente a 1024x768 sube el porcentaje de aciertos. Si seguimos bajando la resolución el porcentaje se reduce progresivamente, pero se mantiene por encima de 80.9 %.

Tabla 7.8: Experimento imágenes generadas por computador Vs escáneres (I)

Prueba	Grupo 1	Grupo 2	Resolución 1	Resolución 2	Imágenes por grupo	% Aciertos
1	Imágenes originales generadas por computador	Imágenes originales de escáneres	$\geq$ 1024x768 y $<$ 2Mpx (1600x1200)	$\geq$ 1024x768 y $<$ 2Mpx (1600x1200)	260	82.1782 %
2	Imágenes reducidas generadas por computador	Imágenes reducidas de escáneres	1024x768	1024x768	260	80.2372 %
3	Imágenes reducidas generadas por computador	Imágenes reducidas de escáneres	640x480	640x480	260	82.1569 %
4	Imágenes reducidas generadas por computador	Imágenes reducidas de escáneres	400x300	400x300	260	82.8794 %

7.1.3 Experimentos con Algoritmo para Distinción de Imágenes por Modelo de Cámara

Este algoritmo no está orientado a la distinción de estos grupos de imágenes, pero debido a que toma muchas más características (4 veces más que el algoritmo especifico para ello, 16 contra 61) y tiene en cuenta muchos más datos, queremos probar como funciona con estos grupos de imágenes y ver si es capaz, en algún caso, de superar al algoritmo original para esto.

Para ello volvemos a realizar las pruebas anteriores más importantes y vemos los resultados, comparándolos con el algoritmo anterior. También, se medirá el tiempo de ejecución de cada algoritmo para diferentes entradas.

Experimentos de Imágenes Escaneadas vs Móviles

Repetimos algunos de los experimentos realizados con el otro algoritmo. Vemos en la Tabla 7.10 que claramente, aún siendo un algoritmo orientado a otro tipo de distinciones,

Tabla 7.9: Experimento imágenes generadas por computador Vs escáneres (II)

Prueba	Grupo 1	Grupo 2	Resolución 1	Resolución 2	Imágenes por grupo	% Aciertos
5	Imágenes recortadas generadas por computador	Imágenes recortadas de escáneres	1024x768	1024x768	260	82.874 %
6	Imágenes recortadas generadas por computador	Imágenes recortadas de escáneres	640x480	640x480	260	81.1024 %
7	Imágenes recortadas generadas por computador	Imágenes recortadas de escáneres	400x300	400x300	260	80.9145 %

es notoria su capacidad para distinguir los grupos, superando al algoritmo original y siendo mucho más estable a diferentes resoluciones.

Inicialmente con las imágenes originales supera al algoritmo anterior en aproximadamente un 1 % pero cuando vamos recortando las imágenes a 400x300 lo supera en casi un 8 %.

Cuando reducimos imágenes en vez de recortarlas vemos que ocurre lo mismo pero en menor medida quedándose en 400x300 el actual algoritmo un 0.5 % por encima del anterior (ver Tabla 7.11).

Otros Experimentos, Móviles vs Generadas por Ordenador y Escaneadas vs Generadas por Ordenador

Repetimos algunos de los experimentos realizados en estas comparaciones.

En el algoritmo anterior al no ser específico para estos grupos de imágenes los porcentajes de acierto eran muy bajos teniendo un máximo en 83 % con todas originales. Con el nuevo algoritmo conseguimos un mínimo de un 85 % con imágenes recortadas a 400x300.

Aún siendo mejor que el algoritmo de distinción de imágenes y cámaras no funciona tan bien como para el anterior caso (ver Tabla 7.12).

En la distinción de las generadas por ordenador y las de escáneres vemos que este algoritmo funciona algo mejor que para el anterior caso. Mostramos a continuación algunos resultados de experimentos realizados con imágenes recortadas (ver Tabla 7.13).

Tabla 7.10: Experimento de imágenes escaneadas Vs móviles(I)

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
1	Imágenes originales de escáner	Imágenes originales de móviles	1024x768	$\geq$ 1024x768 y $<$ 2Mpx (1600x1200)	260	96.1905 %
2	Imágenes recortadas de escáner	Imágenes recortadas de móviles	1024x768	1024x768	260	95.2381 %
3	Imágenes recortadas de escáner	Imágenes recortadas de móviles	640x480	640x480	260	94.2857 %
4	Imágenes recortadas de escáner	Imágenes recortadas de móviles	400x300	400x300	260	92.9524 %

Tabla 7.11: Experimento imágenes escaneadas Vs móviles (II)

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
5	Imágenes originales de escáner	Imágenes reducidas de móviles	1024x768	1024x768	260	95.8095 %
6	Imágenes reducidas de escáner	Imágenes reducidas de móviles	640x480	640x480	260	93.75 %
7	Imágenes reducidas de escáner	Imágenes reducidas de móviles	400x300	400x300	260	93.5838 %

7.2 Experimentos y resultados con imágenes de móviles

Para comprobar la eficacia del algoritmo que obtiene características para la distinción de los diferentes terminales, se harán diferentes experimentos variando los modelos de teléfono móvil que tenemos y el número de fotografías en estos y así comprobar cómo afecta esto al algoritmo. Se tomará siempre como mínimo, un número de 30 fotografías de cada móvil.

En nuestra primera prueba tomaremos 10 modelos de móvil diferentes y de varias marcas los cuales enumeramos a continuación:

Tabla 7.12: Otros experimentos imágenes generadas por computador Vs móviles

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
1	Imágenes originales generadas por computador	Imágenes originales de móviles	$\geq$ 1024x768 y $<$ 2Mpx (1600x1200)	$\geq$ 1024x768 y $<$ 2Mpx (1600x1200)	260	96.1464 %
2	Imágenes reducidas generadas por computador	Imágenes reducidas de móviles	1024x768	1024x768	260	90.9091 %
3	Imágenes recortadas generadas por computador	Imágenes recortadas de móviles	1024x768	1024x768	260	93.1559 %
4	Imágenes recortadas generadas por computador	Imágenes recortadas de móviles	640x480	640x480	260	89.1429 %
5	Imágenes recortadas generadas por computador	Imágenes recortadas de móviles	400x300	400x300	260	85.3053 %

- Iphone 3G
- Iphone 3Gs
- Blackberry 8520
- HTC Desire HD
- LG Ku990i
- Nokia 5300

Tabla 7.13: Otros experimentos imágenes generadas por computador Vs escáneres

Prueba	**Grupo 1**	**Grupo 2**	**Resolución 1**	**Resolución 2**	**Imágenes por grupo**	**% Aciertos**
6	Imágenes recortadas generadas por computador	Imágenes recortadas de escáneres	1024x768	1024x768	260	94.162 %
7	Imágenes recortadas generadas por computador	Imágenes recortadas de escáneres	640x480	640x480	260	92.6415 %
8	Imágenes recortadas generadas por computador	Imágenes recortadas de escáneres	400x300	400x300	260	91.1153 %

- Nokia 6110
- Nokia N95
- Nokia E61i
- Sony Ericsson W580i

De todos estos modelos tomamos 30 fotografías exactamente.

Los resultados son de un 89.4558 % de acierto. Siendo tan pocas fotografías de cada grupo el resultado parece bastante bueno.

Para probar con más cantidad de fotografías cogemos otro grupo de móviles, de los cuales se van a coger un mínimo de 100 fotografías con un máximo de 182. Serán los siguientes 7 modelos:

- Blackberry 8520
- HTC Desire HD
- LG Ku990i
- Nokia 5300
- Nokia 6110

- Nokia 6300
- Sony Ericsson T707
- Sony Ericsson W580i

El resultado en este caso es de un 94.2227 % de aciertos. Vemos por tanto que el rendimiento del algoritmo es bastante mejor en este caso al haber más imágenes pero también se ve afectado porque tenemos 3 modelos menos.

Para hacer una comparación más directa de cuanto afecta el número de clases al rendimiento, vamos a comparar los modelos comunes de ambos, grupos 5 en total y hacemos un test con 30 fotografías cada uno y luego uno con 100 fotos de cada uno. Los modelos comunes, cada uno de una marca, son:

- Blackberry 8520
- HTC Desire HD
- LG Ku990i
- Nokia 5300
- Sony Ericsson W580i

Para 30 fotos tenemos un porcentaje de aciertos de 95.8621 %.

Con 100 fotos cada grupo tenemos un porcentaje de aciertos de 96.2 %.

Vemos que la diferencia no es tan grande como podríamos esperar, probamos con solo 2 grupos para ver si esa diferencia es aún menor. Nos quedamos con:

- Blackberry 8520
- Sony Ericsson W580i

En la Tabla 7.14 se observa que el porcentaje de aciertos no varía demasiado, incluso desciende con 150 sobre el máximo anterior 96.6667 % con lo que podemos concluir que el número de imágenes no afecta demasiado al porcentaje de acierto en este algoritmo (siempre teniendo un mínimo razonable).

Tabla 7.14: % de Acierto por Número de Fotos

Número de fotos	**% de acierto**
30	96.6667 %
60	95 %
90	94.4444 %
120	96.6667 %
150	96.3333 %

Vamos a probar ahora como afecta el número de clases que tenemos al rendimiento del algoritmo. Tomamos 30 fotos de cada modelo de teléfono, que son los 17 siguientes:

- Iphone 3G
- Iphone 3Gs
- Blackberry 8520
- HTC Desire HD
- LG Ku990i
- Nokia 5300
- Nokia 6110
- Nokia N95
- Nokia E61i
- Sony Ericsson W580i
- HTC TYTN II
- Nokia 6300
- Sony Ericsson T707
- Sony Ericsson Z610i
- Sony Ericsson Xperia Neo V
- Sony Ericsson Xperia Arc
- Samsung Galaxy S2

Hemos añadido a los 10 del principio 7 modelos más. Los resultados se muestran en la Figura 7.1.

Como es de esperar, el porcentaje de aciertos depende del número de clases que tengamos. Obtenemos varios picos en la gráfica, que nos denotan la dificultad de determinados modelos para ser distinguidos y que al ser sacados de las pruebas el porcentaje de aciertos sube considerablemente. Se puede considerar la gráfica casi lineal y que cada 10 clases añadidas el porcentaje de aciertos se reduce en torno a un 7-8 %.

Continuamos con las pruebas, ahora vamos a probar los 2 móviles con más resolución 8 Mpx cada uno, 30 imágenes cada uno.

- Sony Ericsson Xperia Arc
- Samsung Galaxy S2

El resultado de este test es un acierto de 100 %. Al tener fotos mucho más grandes y ser modelos de marcas diferentes nos da facilidades a la hora de distinguirlos.

Probamos ahora los 4 modelos más modernos, cada modelo tiene mínimo 5 Mpx de resolución, los cuáles pensamos que darán un alto porcentaje de aciertos al tener fotos tan grandes. Los modelos son:

- HTC Desire HD
- Sony Ericsson Xperia Neo V

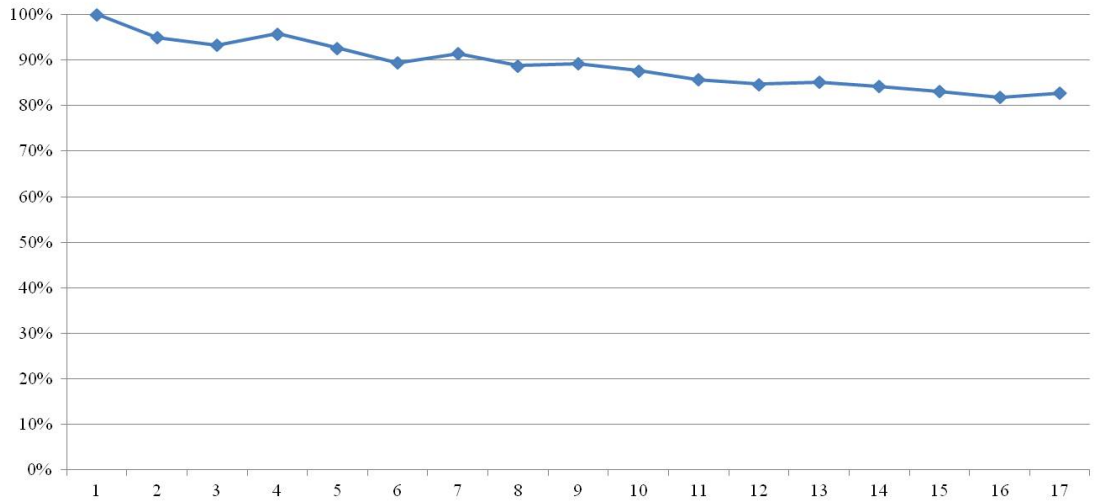


Figura 7.1: Porcentaje de aciertos por número de clases

- Sony Ericsson Xperia Arc
- Samsung Galaxy S2

El porcentaje de aciertos con 4 modelos es de 93.333 %. Vemos que el porcentaje es algo inferior a lo esperado. Como tenemos 2 móviles de la misma marca y bastante parecido quitamos uno. Nos quedamos con todos menos con el Xperia Neo V. El porcentaje de aciertos en este caso es de 97.7778 %.

Si hacemos lo mismo pero eliminando el HTC Desire HD obtenemos un 92.2222 %. Con lo que la culpa estaba en el Xperia Neo V al ser parecido al Xperia Arc. Si eliminamos el Xperia Arc obtenemos un 98.889 % y por último si comparamos el Xperia Neo V con el Arc directamente obtenemos un 95 % que siendo solo 2 modelos, nos es un porcentaje muy alto de aciertos.

Ahora hacemos lo mismo con el test análogo, probamos con móviles más antiguos que tienen entre 3.2 y 2 megapíxeles de resolución. Los modelos son los siguientes:

- Iphone 3g
- BlackBerry 8250
- HTC TYTN II
- Nokia 6300

Obtenemos un porcentaje de aciertos de un 98.3333 % es cuál es un porcentaje muy alto para diferenciar 4 modelos.

Para explicar esto podemos pensar que, aún teniendo peor resolución, estas cámaras son peores y por tanto tienen más defectos e imperfecciones que las caracterizan individualmente. Esto puede ser en gran medida cierto, pero tenemos que tener en cuenta que es

mucho más importante los modelos que tomamos, si son de la misma marca o son modelos parecidos.

Para concluir, vamos a probar un experimento con imágenes de terminales de una sola marca, para ver cómo podría afectar esta característica al rendimiento.

En este caso cogemos todos los terminales de la marca nokia que tenemos, en total 5 modelos.

- Nokia 5300
- Nokia 6110
- Nokia N95
- Nokia E61i
- Nokia 6300

En este caso el porcentaje de aciertos es de un 89.3333 %. Si nos quedamos con los modelos:

- Nokia 5300
- Nokia 6110
- Nokia 6300

Los cuales son los peores modelos, el porcentaje de aciertos para los 3 es de un 100 %. En este caso al ser modelos peores a los usados de 5 y 8 Mpx las imperfecciones hacen que podamos distinguir de forma efectiva los diferentes modelos, aún siendo de la misma marca.

Capítulo 8

Conclusiones y Trabajo Futuro

Después de mostrar los diferentes resultados vemos que conviene en todos los casos usar el algoritmo de 61 características sobre el que tiene solo 16 específicas del ruido. El uso de este algoritmo tiene una contrapartida y es su elevado tiempo de ejecución como mostramos en la Tabla 8.1:

Tabla 8.1: Tiempos de ejecución de los Algoritmos

Prueba	**Resolución**	**Número de imágenes**	**Tiempo algoritmo Ruido (16)**	**Tiempo algoritmo Genérico (61)**	**Proporción en tiempo**
1	1024x768	520	76.63”	953.07”	12,43
2	640x480	520	30.81”	368.13”	11.94
3	400x300	520	19.45”	123.207”	6.3345

Estos tiempos son los resultantes de la ejecución solo de la parte de obtención de características, eliminando la parte de la clasificación por el SVM. El algoritmo, en estos casos, está paralelizado en 4 hilos de ejecución y ejecutado en un equipo con las siguientes características principales:

- Procesador Intel Core I5-460M a 2.56 GHz con 2 núcleos y 4 hilos de ejecución simultáneos
- 4 GB de memoria Ram DDR3

Como podemos observar a mayor resolución, mayor es el tiempo del algoritmo genérico de 61 características en proporción con el algoritmo de características de ruido. Vemos que con imágenes de 1024x768 tarda más de 12 veces mientras que a 400x300 se reduce a 6.33 veces. Por lo tanto, para imágenes pequeñas el algoritmo de 61 características es idóneo, mientras que para imágenes grandes puede ser impracticable por su elevado coste en tiempo.

El uso de estos algoritmos sobre los diferentes tipos de imágenes para la identificación de la fuente, nos da una serie de conclusiones que detallamos a continuación.

En primer lugar vemos algo común a ambos algoritmos, que es característico del SVM y de todos los clasificadores en general y es que los conjuntos de imágenes han de tener un valor mínimo para el entrenamiento óptimo del SVM. Cuantos más ejemplos tengamos de una determinada clase de imágenes, mejor se comportará para imágenes desconocidas, menos fallará y será más tolerante a imágenes mal clasificadas (ruido). Si no hacemos caso a esto, en la mayoría de casos, nuestra fase de entrenamiento nos estará dando resultados erróneos de capacidad de clasificación, que no tienen mucho que ver con la eficiencia real del sistema.

En cuanto a los algoritmos vemos que cada uno es recomendable para determinados casos, siendo uno muy rápido pero poco efectivo a la hora de clasificar imágenes y el otro muy efectivo, pero impracticable en tiempo para muchas imágenes de tamaños grandes y con una capacidad de computación baja. Según el caso que queramos analizar nos compensará más usar uno u otro.

El algoritmo de distinción de imágenes por correlaciones sobre patrones de ruido funciona bien entre imágenes escaneadas e imágenes tomadas desde una cámara fotográfica, ya sea de móvil o una cámara digital normal. También, este algoritmo funciona correctamente para las combinaciones de imágenes escaneadas y generadas por computador e imágenes tomadas con móviles y generadas por computador, pero en estos casos funciona mucho peor.

El otro algoritmo funciona bien en la mayoría de los casos, incluso superando en imágenes escaneadas contra imágenes tomadas desde teléfono móvil al algoritmo especifico para ello. Este algoritmo es muy tolerable a redimensionamientos de las imágenes y por tanto clasificará correctamente imágenes iguales de diferentes resoluciones.

La contrapartida de este segundo algoritmo es su tiempo de ejecución, el cual es muy superior al del primer algoritmo y que cuanto más grandes sean las imágenes que queramos analizar, mayor será esta diferencia de tiempos entre los algoritmos.

8.1 Trabajos Futuros

Las posibles ampliaciones a los algoritmos desarrollados en este proyecto podrían ser la selección de un grupo de características específico que mejorase la tasa de acierto del algoritmo, así como la inclusión de otras características como las basadas en la aberración de lentes. Otra línea de investigación se podría basar en el análisis de la fuente de un vídeo grabado con una cámara de móvil o incluso grabaciones de sonido hechas con estos u otros dispositivos.

Bibliografía

[BHvR03] R. Barr, Z. J. Haas, and R. van Renesse. *Introducción a los Clasificadores*, chapter 5, pages 167–202. 2003.

[BIC] Background Information on CCD and CMOS Technology, http://www.tedpella.com/cameras_html/ccd_cmos.htm.

[CCD] Sensores con tecnología CCD vs CMOS, http://www.xatakafoto.com/camaras/sensores-con-tecnologia-ccd-vs-cmos.

[CFA] Color filter array, http://www.Dpreview.com.

[CLW06] K.-S. Choi, E. Y. Lam, and K. Y. Wong. Automatic source camera identification using the intrinsic lens radial distortion. *Optics Express*, 14(24):11551–11565, November 2006.

[CSA08] O. Celiktutan, B. Sankur, and I. Avcibas. Blind identification of source cell-phone model. *IEEE Transactions on Information Forensics and Security*, 3(3):553–566, September 2008.

[DSM07] A. E. Dirik, H. T. Sencar, and N. Memon. Source camera identification based on sensor dust characteristics. In *Proceedings of the IEEE Workshop on Signal Processing Applications for Public Security and Forensics, 2007 (SAFE '07)*, pages 1–6, April 2007.

[GBK+01] Z. Geradts, J. Bijhold, M. Kieft, K. Kurosawa, K. Kuroki, and N. Saitoh. Methods for identification of images acquired with digital cameras. In *Proceedings of the Enabling Technologies for Law Enforcement and Security*, volume 4232, pages 505–512, February 2001.

[IDG] Informativo IDG, http://www.idg.es/dealerworld/Las-ventas-de-telefonos-moviles-creceran-un-pobre-4-por-ciento-en-2012/seccion-mercado/noticia-122806.

[KKS99] K. Kurosawa, K. Kuroki, and N. Saitoh. Ccd fingerprint method-identification of a video camera from videotaped images. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 3, pages 537–540, 1999.

[KMC+06] N. Khannaa, A. K. Mikkilinenib, G. T. Chiub, J. P. Allebacha, and E. J. Delpa. Forensic Classification of Imaging Sensor Types. Rfc, Purdue University, February 2006.

[LFG06] J. Lukas, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*, 1(2):205–214, June 2006.

[LP05] R. Lukac and K.N. Plataniotis. Color filter arrays: design and performance analysis. *IEEE Transactions on Consumer Electronics*, 51(4):1260–1267, November 2005.

[MSM04] K.L. Mehdi, H.T. Sencar, and N. Memon. Blind source camera identification. In *Proceedings of the International Conference on Image Processing (ICIP '04)*, volume 1, pages 70–712, October 2004.

[Nak05] J. Nakamura. *Image Sensors and Signal Processing for Digital Still Cameras*. CRC Press, Inc., Boca Raton, FL, USA, 2005.

[PAR] Parallel Python, http://www.parallelpython.com/.

[RSYD05] R. Ramanath, W.E. Snyder, Y. Yoo, and M.S. Drew. Color image processing pipeline. *IEEE Signal Processing Magazine*, 22(1):34–43, January 2005.

[SNAPO07] Z. L. Sandoval NiÃand F. A. Prieto Ortiz. Caracterización de Café Cereza Empleando Técnicas de Visión Artificial. *Revista Facultad Nacional de Agronomía, Medellín*, 60:4105 – 4127, 12 2007.

[SWL07] A. Swaminathan, Min Wu, and K.J.R. Liu. Nonintrusive component forensics of visual sensors using output images. *IEEE Transactions on Information Forensics and Security*, 2(1):91–106, March 2007.