UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA INFORMÁTICA

29/11/2011

DISEÑO, ANÁLISIS Y OPTIMIZACIÓN DE UN SISTEMA DE RECONOCIMIENTO DE IMÁGENES BASADAS EN CONTENIDO PARA IMAGEN PUBLICITARIA

PROYECTO FIN DE CARRERA

AUTOR:
Javier Juan Albarracín

TUTORES:
Juan Miguel García Gómez

Elies Fuster i Garcia

Jose Luis Bayo Montón

Agradecimientos

En primer lugar, quiero agradecer enormemente a Juan Miguel García Gómez la oportunidad que me ha dado de formar parte del grupo IBIME y abrirme las puertas a esta nueva etapa de mi vida. Gracias por apostar por mí para formar parte del grupo.

En segundo lugar quiero agradecer a Elies Fuster i Garcia la inestimable ayuda que me ha prestado, tanto a nivel profesional para llevar a cabo este proyecto, como a nivel personal para desarrollarme y crecer como persona.

Quiero agradecer a todos los compañeros del grupo de minería de datos y *pattern recognition* de IBIME por la ayuda prestada y por haberme acogido como uno más desde el primer día. Gracias a Salva, Carlos, Adrián, Miguel, Javi, Alfonso y Montserrat Robles, la directora del grupo, por hacerme más fácil este camino.

A mis amigos de la facultad Alex Pérez, Alex Temina, Héctor Yone y todos los demás por acompañarme en este viaje y compartir tan buenos momentos.

Gracias también a mis padres Rafael y María Ángeles y a mi hermano Eduardo, por la formación que me han dado en la vida. Sin los valores de trabajo y esfuerzo que me han inculcado desde pequeño no habría llegado a donde estoy hoy.

Y por último y la más importante, gracias a ti Itziar por todo el apoyo incondicional que me das cada día. Gracias por estar siempre a mi lado, sacar lo mejor de mí y por cambiarme la vida. Sin ti nada de esto habría sido posible.

A todos vosotros va dedicado este proyecto con el que finalizo mi carrera, que es para mí la mejor de las herencias. Muchas gracias.

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

Hoy en día, la publicidad es un sector en continuo crecimiento, tanto por la repercusión de su actividad en la sociedad, como por la importancia que supone para el desarrollo económico de una empresa. El éxito de un negocio comienza por una buena difusión de sus productos a los clientes y pasa por mantenerse a la vanguardia de los nuevos métodos publicitarios tanto en diseño, como en estudio de la influencia de los mismos sobre la sociedad. Es por lo tanto necesario, cuantificar económicamente la repercusión de la publicidad sobre los beneficios obtenidos por una empresa, puesto que esta puede traducirse por supuesto en grandes sumas de dinero, pero también en mayores cantidades de beneficio para la empresa. El *clipping* es la actividad encargada de la cuantificación y análisis económico de la repercusión de la publicidad sobre la sociedad. El *clipping* consiste en la obtención de aquellas noticias de interés para la empresa, y su posterior análisis para comprobar el impacto que puedan suponer sobre el público objetivo de la noticia. La automatización del proceso de *clipping* es un objetivo que se lleva persiguiendo desde hace tiempo, debido a su naturaleza tediosa y repetitiva, y también al elevado índice de coste que implica su consecución manual.

Una forma posible de medir este impacto es a través del número de apariciones del logotipo de la empresa en estos medios. Las técnicas de reconocimiento de imágenes pueden contribuir substancialmente a cuantificar este valor de forma precisa. Los sistemas de reconocimiento de imágenes son ampliamente conocidos y aplicados en diferentes áreas. En la actualidad existen sistemas de reconocimiento de imágenes aplicados a rostros, huellas dactilares, *tracking* de objetos en movimiento, etc. pero su desarrollo para la detección de publicidad no está siendo ampliamente explotado. La necesidad de sistemas de estas características se hace cada vez más fuerte a la hora de evaluar el enorme volumen de información que se dispone hoy en día.

Este proyecto forma parte del proyecto REIMED (Investigación y desarrollo de nuevos procesos inteligentes de reconocimiento e impacto en medios de comunicación), cuyo reto tecnológico consiste en desarrollar un sistema integral de información de inteligencia mediática. En este proyecto se colabora con la empresa *Auditoria de Medios* (en adelante *AuditMedia*), que es una compañía valenciana dedicada al *clipping* y coordinadora de dicho proyecto. El objetivo principal del proyecto REIMED consiste en la investigación y desarrollo de nuevos procesos inteligentes de reconocimiento de información, y la consecución de un modelo de análisis y predicción del impacto que las noticias e información publicada sobre una empresa tiene en la evolución de su cotización bursátil. Se pretende que los nuevos procesos de reconocimiento de información supongan una mejora sustancial, se estima que entre un 60% y un 80%, en la eficiencia y productividad de la empresa. El proyecto consta de tres líneas de trabajo:

1. Desarrollo de un sistema avanzado de reconocimiento inteligente del habla – RIH
2. Desarrollo de un novedoso sistema de reconocimiento inteligente de formas aplicado a logotipos – RIF
3. Desarrollo de un nuevo sistema de predictivo del impacto que la información publicada en los medios sobre una compañía tiene en su cotización bursátil – SPI

Este proyecto cubre la segunda línea de trabajo del proyecto REIMED. En este estudio se ha propuesto el diseño, análisis y optimización de un sistema de reconocimiento de formas, y en concreto de una herramienta de reconocimiento de imágenes para la ayuda a la evaluación del impacto publicitario de una entidad, en los medios de comunicación. Actualmente *AuditMedia* no presta el servicio de reconocimiento y evaluación de logotipos a sus clientes, por lo que la incorporación de un

sistema de estas características supone un valor añadido y una diferenciación de los principales competidores del sector.

1.2 OBJETIVOS

El objetivo del sistema que se presenta en este proyecto consiste en construir una herramienta potente y fiable de ayuda a la evaluación del impacto publicitario de una marca en diferentes medios de comunicación. Más concretamente el sistema a diseñar debe ser capaz de localizar anuncios de una empresa cliente en los diferentes medios de difusión de prensa escrita, robusta frente a distorsiones de las imágenes, cambios de contrastes, ocultación parcial, orientación, tamaño, etc. En este proyecto, se aborda únicamente la detección de publicidad en imágenes estáticas, sin embargo el proyecto esta embebido en un sistema más complejo cuyo objetivo es la detección de publicidad tanto en imagen estática, como en imagen dinámica y audio, es decir, en video, televisión, radio, etc.

Los objetivos específicos que se pretenden conseguir en este proyecto son:

1. Estudio del estado del arte en técnicas de reconocimiento de formas aplicadas a la detección automática de logotipos y marcas.
2. Estudio de viabilidad del sistema de detección automática de logotipos y marcas mediante el diseño de un sistema de reconocimiento de patrones.
3. Diseño del sistema de reconocimiento de formas para la detección automática de logotipos y marcas.
4. Implementación de un prototipo experimental.
5. Evaluación de los resultados obtenidos y selección de modelos óptimos.
6. Implementación y optimización de una librería final de carácter comercial.
7. Interpretación de los resultados.

1.3 ESTADO DEL ARTE

El sistema de detección automática de logotipos y marcas que se pretende diseñar en este proyecto, se engloba en los sistemas denominados: *Sistemas de detección de imágenes basadas en contenido (CBIR: Content Based Image Retrieval)* [1], o en ocasiones también llamados sistemas de *consulta de imágenes por contenido (QBIC: Query By Image Content)* [2]. El objetivo principal de estos sistemas consiste en la búsqueda de imágenes o fragmentos de imágenes, en grandes bases de datos. Como su nombre indica, la búsqueda se realiza basándose en el contenido de la imagen y no en los *metadatos* que puedan llevar asociados dichas imágenes. El término *contenido* hace referencia pues a las formas, colores o figuras que definen la imagen.

Los sistemas *CBIR* presentan diferentes técnicas de consulta. Principalmente estas consultas se dividen en dos grandes grupos: consultas mediante ejemplo y consultas mediante información semántica. El primer tipo de consultas es el más común y el empleado en este proyecto. El usuario del sistema elije o proporciona una imagen como patrón de ejemplo, y el sistema debe comparar dicha imagen contra toda la base de datos para devolver aquellas imágenes similares al patrón de entrada. Por otro lado, los sistemas basados en la semántica de la consulta reciben una oración sobre lo que el usuario desea encontrar y el sistema debe proporcionar como salida aquellas imágenes que se ajusten a la petición. El sistema diseñado en este proyecto hace uso de la búsqueda mediante patrón de ejemplo.

El método más común de comparación del contenido de una imagen es el basado en distancias. Los sistemas *CBIR* evalúan la similitud de dos imágenes en función de sus distancias basados en diferentes medidas, como por ejemplo color, textura, formas, etc. Es por tanto necesario, hacer uso de

métodos de extracción de características que sean capaces de extraer descriptores que sinteticen la mayor parte de la información útil para la detección de la imagen. Estos algoritmos tratan de describir la imagen a través de un conjunto de puntos concretos de la misma y mediante la información espacial de cada punto [3]. Actualmente, los algoritmos basados en esta aproximación son los más utilizados gracias a su buen rendimiento.

Uno de los detectores más populares y que más se ha utilizado es el *Harris corner detector* [4], propuesto en 1988 por *C. Harris* y *M. Stephens*. Sin embargo, este detector no era originalmente invariante a escala por lo que terminó siendo desbancado por otros algoritmos más potentes. *Lindeberg* [5] introdujo en 1998 el concepto de selección automática de escala y revolucionó el desarrollo de los algoritmos de extracción de características. Esta nueva aproximación permitía extraer puntos interesantes que describieran las imágenes, independientemente de la escala a la que se encontrara la misma. *Mikolajczyk* y *Schmid* [6] refinaron el trabajo de *Lindeberg* y presentaron un método robusto e invariante a escala llamado *Harris-Laplace and Hessian-Laplace detector*, que hacía uso del operador laplaciano, en concreto del operador *LoG (Laplacian of Gaussian),* como herramienta para la detección de puntos interesantes*. David Lowe* [7] hizo una contribución importantísima para los algoritmos actuales, proponiendo aproximar el operador *LoG* mediante un filtro de diferencia de gaussianas, *DoG (Difference of Gaussian)* cuyo coste es muchísimo menor y por lo tanto muchísimo más rápido. Esta aproximación dio como resultado uno de los métodos más importantes en el reconocimiento de imágenes llamado *SIFT (Scale Invariant Feature Transform)*. Este método se considera el precursor de los métodos actuales de extracción de características. Posteriormente, se han construido diferentes algoritmos basados en la filosofía *SIFT* que mejoran algunos aspectos en la detección de patrones en imágenes. En concreto, uno de los sucesores más importantes de *SIFT,* ha sido el algoritmo *Speeded-Up Robust Features (SURF)* [8]. *SURF* fue presentado en 2006 en el ECCV en Graz (Austria). Está parcialmente inspirado en *SIFT* y se ha demostrado que en la práctica totalidad de los casos consigue mejorar el rendimiento de este algoritmo [9]. Se basa en el cálculo del determinante de la matriz *Hessiana (DoH: Determinant of Hessian)* para la detección de puntos interesantes y en las *wavelets* de *Haar* para la descripción de dichos puntos. Esta aproximación es aún más rápida que *DoG* y ofrece una respuesta superior en cuanto a calidad de descripción de las imágenes.

Los algoritmos SIFT y *SURF* son algoritmos de amplia difusión e implementados en muchas bibliotecas de procesamiento de imágenes. Ofrecen un rendimiento y velocidad excelentes así como unas cualidades de repetibilidad y precisión elevadas. Así mismo *D. Badanov*, *Lamberto Ballan, Marco Bertini, Alberto del Bimbo y Arjun Jain* presentan en sus trabajos [10] y [11] un sistema de reconocimiento de logotipos orientado a video basado en descriptores SIFT similar al que se presenta en este proyecto.

1.4 Introducción a la Inteligencia artificial y el Reconocimiento de formas en imágenes digitales

1.4.1 Inteligencia Artificial. Reconocimiento de formas

La *Inteligencia Artificial (IA)* se puede definir como la habilidad de una máquina para llevar a cabo tareas que normalmente se asocian con la inteligencia humana, tales como el razonamiento o el aprendizaje. La *IA* se divide en dos grandes ramas: la *Inteligencia Artificial Clásica* y la *Inteligencia Artificial Moderna.* El enfoque que se ha adoptado en este estudio es el enfoque moderno.

La *Inteligencia Artificial Clásica* centra sus esfuerzos en aproximarse al razonamiento humano, el aspecto cognitivo de la inteligencia y la inferencia deductiva de conocimiento. El marco principal en el que se desarrollan estos sistemas es la lógica. El enfoque de la IA clásica utiliza técnicas de aprendizaje

deductivo mediante las cuales un humano experto transfiere conocimiento al sistema y este aplica dicho conocimiento para la resolución del problema. Consecuentemente, el sistema infiere más conocimiento y amplía así su base de reglas aprendiendo nuevos predicados. Los sistemas más comunes que utilizan esta aproximación son los llamados *Sistemas Expertos (SE)*, *Sistemas Basados en Conocimiento (SBC)*, *Sistemas Multi-Agente (SMA), agentes inteligentes* o *sistemas de satisfacción de restricciones (CSP: Constraint Satisfaction Problems)*.

El enfoque de la *Inteligencia Artificial Moderna* es diferente. Se basa en la percepción del entorno, la información sensorial y la interacción con el medio. El reconocimiento de formas se sitúa dentro de esta aproximación y se define como la ciencia que se encarga de desarrollar sistemas informáticos capaces de reconocer su entorno mediante información adquirida por múltiples sensores, por ejemplo: ópticos, acústicos, térmicos, etc. El marco natural sobre el que se construye la IA moderna es la estadística y la teoría de la decisión. Los sistemas desarrollados bajo este enfoque basan sus técnicas en el aprendizaje inductivo, extrayendo conocimiento a partir de ejemplos proporcionados como *input* del sistema, y a través de un proceso de entrenamiento. Algunos objetivos específicos del reconocimiento de formas son: el reconocimiento de imágenes, reconocimiento del habla, reconocimiento de escritura, procesamiento del lenguaje natural, biometría, medicina, etc.

La forma habitual para abordar problemas de reconocimiento de formas es la *clasificación*. La *clasificación* consiste en la asignación de una *etiqueta* a cada muestra de manera que pase a formar parte de un subconjunto bien conocido de muestras que comparten, todas ellas, características comunes que las definen. Bajo este paradigma, los problemas iniciales propuestos se transforman en tareas de clasificación en las que, dado un conjunto de objetos a clasificar y un grupo de clases posible, el sistema tratará de asignar cada uno de los objetos, con el mínimo error posible, en alguna de las anteriores clases. Formalizando la sentencia anterior el proceso de clasificación se modela como:

$$(objeto)\ x \rightarrow\ [Sistema\ de\ Reconocimiento\ de\ Formas] \rightarrow c(x)\ \epsilon\ \{1, \dots, C\}\ (etiqueta\ de\ clase)$$

En conclusión, un sistema de reconocimiento de formas consiste en los siguientes tres módulos:

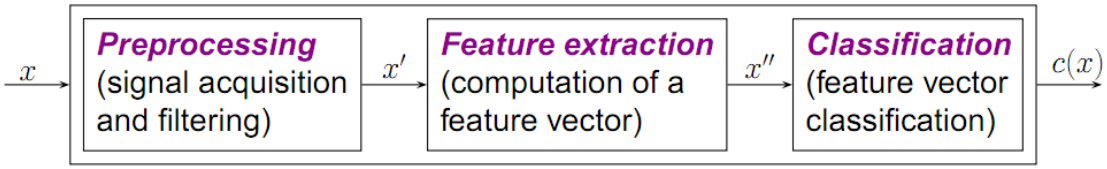

FIGURA 1.1: ESQUEMA DE LAS PARTES PRINCIAPLES DE UN SISTEMA DE RECONOCIMIENTO DE FORMAS

El primer módulo es un módulo de adquisición y pre-proceso de datos. El objetivo principal de este módulo consiste en el registro de los datos de entrada y su preparación para facilitar al máximo la tarea de clasificación y obtener así los resultados óptimos. Este módulo, es de gran importancia sobre todo cuando los datos de entrada al sistema albergan ruido o información confusa que debe ser filtrada o eliminada. Para nuestro caso de estudio, las técnicas de pre-proceso que aplicamos son técnicas de tratamiento de imágenes. Algunas de estas técnicas o herramientas que se han utilizado en este estudio son: técnicas sobre píxel, técnicas de gestión de la información de color, filtros y ajuste de información espacial. En el capítulo 3.2.1 se profundizará en el Pre-Proceso y se detallarán dichas técnicas.

El módulo de extracción de características es el encargado de definir los rasgos principales de cada clase y las características que las diferencian. Su objetivo principal es dar una representación compacta y normalizada de los datos con la mayor información discriminante posible. Esta representación debe darse en un formato óptimo para la máquina por lo que la representación típica

suele ser en formato vectorial. Así pues, esta tarea es crítica ya que es necesario definir de forma exacta y precisa la información que disponemos para crear modelos ajustados de la realidad que sirvan como referencia en el proceso de clasificación. Es imposible diseñar un sistema de reconocimiento de formas fiable sin diseñar un módulo de extracción de características robusto que describa la información de forma óptima. En el caso de estudio que se presenta, se ha seleccionado *Speeded-Up Robust Features (SURF)* y *Phase Correlation* como aproximaciones al problema. En el capítulo 3.2.2 se profundizará en ambos métodos.

Por último, el módulo de clasificación es el encargado de asignar una etiqueta de pertenencia a una determinada clase, a cada muestra. Para ello, previamente es necesario *entrenar* al clasificador, es decir, *enseñar* mediante ejemplos cuáles son las características fundamentales que distinguen cada clase. Existen principalmente dos técnicas de aprendizaje en el reconocimiento de formas: el *aprendizaje supervisado* y el *aprendizaje no supervisado*. Cada uno de estos métodos de aprendizaje y sus correspondientes técnicas de clasificación serán desarrollados en profundidad en el capítulo 3.2.3. En concreto, en este proyecto se han usado técnicas de *aprendizaje supervisado,* y clasificadores *LDA/QDA (Linear/Quadratic Discriminant Analysis)*.

1.4.2 IMÁGENES DIGITALES

En el ámbito informático, una *imagen digital* es la representación numérica, normalmente binaria, de una imagen principalmente bidimensional. Aunque existen imágenes tridimensionales, estas no participan en este caso de estudio y no se comentarán puesto que se salen de los objetivos de este proyecto.

Las imágenes digitales se pueden dividir en dos grandes bloques: las *imágenes rasterizadas* y las *imágenes vectoriales*. Estos dos grandes bloques se diferencian principalmente en el método de almacenamiento de la información. Más en concreto, dependiendo de si la resolución es fija o variable, las imágenes pertenecen al grupo de *imágenes rasterizadas* o *imágenes vectoriales* respectivamente.

Las *imágenes rasterizadas* son más comúnmente llamadas *mapas de bits (bitmaps)* o imagen matricial. Son una representación de la imagen original mediante una rejilla o *grid* de píxeles denominada *raster*, donde cada píxel toma un determinado valor de color. Estas imágenes se suelen caracterizar por su altura, anchura y por la profundidad de color que determina la calidad de la representación de la imagen. Este tipo de imágenes se almacenan en ficheros o estructuras de datos, en las que se guarda el color de cada uno de los píxeles de la imagen para su posterior representación. La principal ventaja de este método es la sencillez y facilidad de gestión de las imágenes, sin embargo el problema que presentan las imágenes *rasterizadas* es la dependencia de la resolución y el escalado para una buena representación.

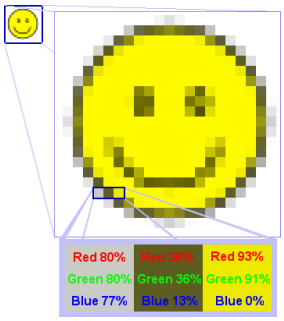

FIGURA 1.2: ESCALADO DE IMAGEN RASTERIZADA

No es posible aumentar o disminuir la resolución de estas imágenes sin perder calidad de las mismas. Es por tanto este factor el más importante a tener en cuenta cuando se opera con este tipo de imágenes. Si un sistema de reconocimiento de formas trabaja con *imágenes rasterizadas*, es imprescindible que estas se encuentren a la mayor resolución posible. Más aún, cuando el objetivo del sistema de reconocimiento consiste en describir la información contenida en cada imagen, este factor es crucial. Si dos imágenes iguales no se encuentran a un nivel de resolución similar, es muy posible que un algoritmo de extracción de características describa estas imágenes de forma muy diferente. Es fácil, como se muestra en la Figura 1.3, que al comparar ambas imágenes, no se asemejen en gran medida.

FIGURA 1.3: RESOLUCIÓN EN IMÁGENES RASTERIZADAS

Lamentablemente, las imágenes con las que cuenta el sistema de reconocimiento de logotipos, son imágenes *rasterizadas* y comprimidas con el algoritmo JPEG. Existen multitud de algoritmos de compresión y formato de imágenes. Uno de los más conocidos es el algoritmo JPEG (*Joint Photographic Experts Group)* http://www.jpeg.org/jpeg/index.html, cuyo funcionamiento no se explicará puesto que no corresponde con los objetivos de esta memoria. Sin embargo citar que este algoritmo es un algoritmo de compresión con pérdidas. Esto implica que durante el proceso de compresión se pierde información generalmente asociada con la calidad de la imagen. Por otro lado, el algoritmo también introduce ciertos elementos llamados *artefactos* que distorsionan la imagen e introducen ruido que debe ser filtrado. En el capítulo 3.2.1 apartado 3.2.1.2.2 Filtro denoising, se presentará una posible técnica para eliminar *artefactos.*

Por otro lado se encuentran las *imágenes vectoriales*. La diferencia fundamental entre estas imágenes y las *imágenes rasterizadas* reside en el tipo de información que almacena cada una de ellas. Como se ha comentado anteriormente, las imágenes *rasterizadas* guardan un mapa de los colores de cada uno de los píxeles que forman la imagen. Sin embargo, las *imágenes vectoriales* guardan información, mediante fórmulas matemáticas, de las figuras geométricas que se encuentran en dicha imagen, como por ejemplo líneas, puntos, curvas, polígonos o figuras. Es posible, por tanto, representar posteriormente la imagen, reconstruyéndola a partir de la información almacenada sin que se vea afectada la resolución. Por ejemplo, dada una imagen formada por un circulo de radio *r* y de color rojo, una *imagen vectorial* almacenaría una etiqueta indicando que en la imagen aparece un círculo. Guardaría la posición relativa del punto central del círculo y su radio, y por último el color de la curva que define al círculo y su relleno. Con esta información, es posible representar la imagen a cualquier resolución y escala imaginable sin perder absolutamente ni un ápice de calidad de la misma. Por el contrario, una construcción tan sencilla como esta, sí se vería afectada en una imagen *rasterizada.*

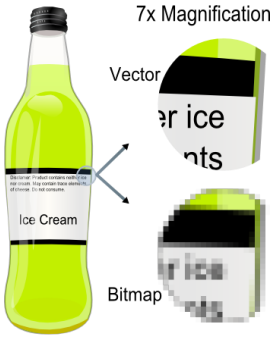

FIGURA 1.4: IMAGEN VECTORIAL

Las primitivas con las que cuentan las imágenes *vectoriales* para definir los elementos contenidos en las imágenes son: líneas, polylineas, polígonos, curvas *Bézier, bezígonos,* círculos y elipses. Estas imágenes permiten también transformaciones afines como rotaciones, movimientos, *mirroring, stretching*, así como operaciones de unión, concatenación, diferencia, intersección, etc.

La complejidad de descripción de estas imágenes ha ido en detrimento de su popularidad por lo que el ámbito natural en el que se opera con este tipo de imágenes, se reduce a áreas más profesionales de desarrollo y tratamiento de imagen. Entre estas áreas se encuentra por ejemplo el diseño gráfico, arquitectura, animación, etc.

En conclusión, debido a la información proporcionada por la empresa destinataria del proyecto, el sistema de detección de logotipos y marcas hace uso de imágenes *rasterizadas* comprimidas en JPEG.

1.4.3 EXTRACCIÓN DE CARACTERÍSTICAS

Uno de los principales problemas a los que se enfrenta el reconocimiento de formas, es la extracción de características o de información relevante de los datos que se desean clasificar. Una tarea tan sencilla para los humanos como la lectura o la escritura, se convierte en una difícil tarea para una máquina. En este punto, se hacen necesarias técnicas de extracción de características que sean capaces de describir con calidad y de forma eficiente la información que disponemos. Más aun, este paso se considera crítico en tanto y cuanto la información que disponemos sea limitada o escasa como ocurre en la mayoría de casos en el ámbito real. La extracción de características es fundamental a la hora de diseñar un sistema de reconocimiento de formas fiable. Es imposible ofrecer unos resultados precisos y de calidad si no se describe la información de forma correcta. Esto es obvio y fácilmente comprensible ya que, cómo vamos a ser capaces de decidir si un objeto pertenece a una determinada clase, sin saber definir las propiedades o características fundamentales de cada una de dichas clases o de los propios objetos.

El objetivo de la extracción de características es dar una representación compacta y normalizada de los datos, conservando la máxima información discriminante posible. Un pequeño ejemplo de extracción de características para imágenes de tamaño 5x5 que representan dígitos numéricos, sería tomar los 25 pixeles que forman cada imagen como 25 variables con las que describir el dígito.

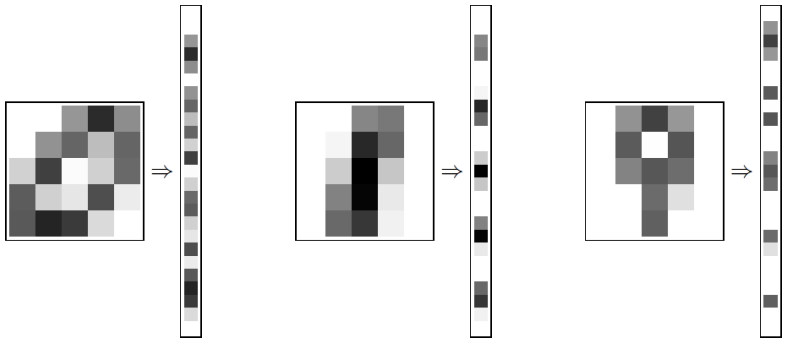

FIGURA 1.5: EJEMPLO DE EXTRACCIÓN DE CARACTERÍSTICAS

Se ve rápidamente que, en general, ésta no es una buena técnica de extracción de características. En el momento en el que procesáramos imágenes de tamaño real, como por ejemplo imágenes de 2300×3000, el número de características que manejaríamos para cada imagen sería inmenso, del orden de 6900000 características. A priori, se puede pensar que cuanto mayor número de características dispongamos, mayor será el grado de discriminabilidad o mayor será la cantidad de información útil que dispongamos, sin embargo esto no es siempre cierto. En el ejemplo anterior existe información irrelevante, como el fondo sobre el cual están los dígitos, que incrementa notablemente el número de características y no aporta ninguna información útil al proceso. Incluso, en ocasiones, puede llegar a introducir información no deseable que llegue a confundir el verdadero significado de la representación. Así pues, es necesario construir técnicas robustas de extracción de características que encuentren un equilibrio óptimo entre el número de características extraídas para representar un conjunto de datos, y la información relevante contenida en las mismas para resolver el problema. Mediante este razonamiento, podemos ver el proceso de extracción de características como un proceso de reducción de la dimensionalidad inicial de los datos de forma que el resultado sea un conjunto de datos de tamaño inferior y que retenga la mayor información discriminante posible. Esta aproximación es muy importante cuando tratamos con grandes conjuntos de datos, ya que reducir la dimensionalidad de estos implica reducir el coste computacional y temporal a la hora de procesarlos. Si formalizamos la anterior definición, tenemos que: Dado un conjunto de datos $x = (x_1, x_2, \dots, x_D)$ de dimensión *D,* el objetivo consiste en encontrar otra transformación de dicho conjunto, $x' = (x_1, x_2, \dots, x_d)$ de dimensión *d*, donde $d < D$ y x' represente de forma más compacta y precisa la información original. Una de las técnicas más utilizadas y que representa exactamente lo anteriormente definido, es *Principal Component Analysis (PCA)* de la que ya se ha hablado en capítulos anteriores. Otras técnicas igualmente conocidas son *Semidefinite embedding (SME), Independent Component Analysis (ICA), Multifactor dimensionality Reduction (MDR)*, y muchas otras más.

En el ámbito concreto de las imágenes existen multitud de algoritmos de extracción de características. Generalmente estos algoritmos constan de dos partes diferenciadas: la *detección* de puntos de interés y la *descripción* de dichos puntos. Para ambas partes existen variedad de algoritmos que se resumen en las siguientes tablas.

Detectores de características	*Edge*	*Corner*	*Blob*
Canny	X		
Sobel	X		
Harris & Stephan / Plessey	X	X	
SUSAN	X	X	
Shi & Tomasi		X	
Level curve curvature		X	
FAST		X	
Laplaciano de la Gaussiana (LoG)		X	X

Diferencia de la Gaussiana (DoG)	X	X
Determinante de la Hessiana (DoH)	X	X
MSER		X
PCBR		X
Grey-level blobs		X

TABLA 1: DIFERENTES DETECTORES DE PUNTOS CON LOS TIPOS DE CARACTERÍSTICA DETECTADA

Descriptores de características	Detectores de características
SIFT	**Diferencia de la Gaussiana/Laplaciano de la Gaussiana (DoG/LoG)**
SURF	**Determinante de la Hessiana (DoH)**
GLOH	**SIFT + PCA**
HOG	**SIFT + Grid Uniforme + Normalización del contraste**
LESH	**Signal accumulating energy**

TABLA 2: DIFERENTES DESCRIPTORES DE CARACTERÍSTICAS BASADOS EN LOS DETECTORES ANTERIORES

En la primera tabla, se clasifican los algoritmos de detección de características en tres grandes bloques: *edges, cornes, blobs.* Cada uno de estos bloques se corresponde con un concepto abstracto de característica. Los algoritmos clasificados en el grupo *edge*, engloban algoritmos capaces de detectar *bordes* en imágenes, mientras que la clase *corners* y *blobs* se corresponden con la detección de *puntos interesantes* y *regiones interesantes* respectivamente.

En conclusión, la extracción de características es uno de los puntos más importantes en el proceso de clasificación de cualquier conjunto de datos. En concreto, en el campo de las imágenes, la extracción de características se suele definir como la detección y descripción de puntos interesantes que ayuden a describir la imagen de la mejor forma posible. Normalmente uno de los pasos de mayor coste computacional es la extracción de características por lo que este proceso puede convertirse en el *cuello de botella* del sistema. En este proyecto se han empleado dos algoritmos de extracción de características: *Phase Correlation* y *Speeded-Up Robust Features (SURF)*. El algoritmo final elegido que constituye el núcleo del sistema de detección de logotipos y marcas es *SURF*.

1.4.4 CLASIFICACIÓN

La clasificación es el paso final en un sistema de reconocimiento de formas. Este paso consiste en, dada una muestra del conjunto de datos, asignar una etiqueta de clase a esa muestra en base a su grado de similitud o pertenencia con dicha clase.

El clasificador es un procedimiento o algoritmo matemático cuyo objetivo es diferenciar entre agrupaciones de datos con características comunes. Una vez entrenados, utilizan dichos modelos para posteriores clasificaciones de datos no conocidos. Un clasificador se define como:

$$c(x) = \arg\max_{c} g_c(x)$$

donde, para cada clase c, existe una función discriminante $g_c$que define el grado de pertenencia del objeto x a la clase c. Dadas estas funciones discriminantes, el objeto x se clasifica en la clase que presenta un mayor grado de correlación o pertenencia con respecto a la función g_c.

La finalidad del clasificador consiste entonces en dividir el espacio de características en C regiones de decisión diferentes, separadas cada una por sus fronteras de decisión:

$$R_c = \{x:\ g_c(x) > \max_{c' \neq c} g_{c'}(x) \qquad para\ todo\ c$$
$$F_{c,c'} = \{x: g_c(x) = g_{c'}(x) \qquad para\ todo\ c \neq c'$$

El clasificador óptimo es el llamado *clasificador de Bayes*, cuya regla es la siguiente:

$$c^*(x) = \arg\max_{c=1,\ldots,C} p(c|x)$$

y cuyo error, llamado *error de Bayes*, es:

$$p(error) = \int p(error, x)dx = \int p(x)p(error|x)dx$$

donde la probabilidad de error dado x, $p(error|x)$, es:

$$p(error|x) = 1 - \max_c p(c|x)$$

Este es el clasificador óptimo de mínimo error, sin embargo es imposible construir dicho clasificador puesto que los requerimientos para obtener la probabilidad condicional $p(c|x)$ son muy difíciles de conseguir y normalmente desconocidos. La regla de *Bayes* nos muestra cómo obtener esta probabilidad condicional:

$$p(c|x) = \frac{p(c)p(x|c)}{p(x)}$$

Esta regla nos indica que para obtener la probabilidad que buscamos necesitamos: la probabilidad a priori de cada clase, $p(c)$, la probabilidad de que dada una clase concreta un objeto pertenezca a dicha clase, $p(x|c)$ y por último la probabilidad general de que un objeto llegue al sistema, $p(x)$. Esta cantidad de información no es fácilmente deducible y en ocasiones es prácticamente imposible de estimar. Por este motivo en la práctica, la implementación de este clasificador es inviable y se toma únicamente como modelo de referencia.

Por otro lado, dada la definición de un clasificador y su naturaleza estadística, la construcción del mismo conlleva un paso de entrenamiento y aprendizaje para obtener las fronteras de decisión entre las diferentes clases. Este proceso de aprendizaje se divide en dos tipos: aprendizaje *supervisado* y aprendizaje *no supervisado*.

El *aprendizaje supervisado* es el más común y se diferencia del *no supervisado* en que en el primero es necesario disponer de las muestras correctamente etiquetadas. En otras palabras, cada muestra consiste en un par formado por el propio dato, y su valor deseado de salida del clasificador. Conociendo a priori información sobre cada muestra, se entrena al clasificador proporcionándole datos bien conocidos (etiquetados), para que cree los modelos discriminantes pertenecientes a cada clase. Los clasificadores más comunes que trabajan con *aprendizaje supervisado* son los clasificadores Bayesianos, redes neuronales, análisis discriminante y máquinas de soporte vectorial (*SVM: Support Vector Machines*). El *aprendizaje supervisado* ha sido la técnica empleada en este caso de estudio.

Por el contrario, en el caso del *aprendizaje no supervisado*, el clasificador no tiene ninguna información a priori de las clases. El clasificador busca agrupaciones de datos *naturales* que compartan características comunes, y asocia a estos conjuntos una etiqueta *sintética* o ficticia para diferenciarlos. Las técnicas de *aprendizaje no supervisado* más comunes son el *clustering, mapas auto-organizativos* y mapas de *Kohonen*.

En la técnica de *aprendizaje supervisado,* el proceso de *entrenamiento* consta de varias fases: En primer lugar se construyen dos grupos o bloques de muestras llamados *conjunto de entrenamiento* y *conjunto de evaluación* o *test*. El *conjunto de entrenamiento* normalmente debe ser mayor en número de muestras que el *conjunto de test*, y se utiliza para el aprendizaje inductivo. El clasificador toma dicho

conjunto de *entrenamiento* y crea un modelo discriminante de los datos, es decir unas fronteras de decisión entre las clases que utiliza posteriormente para la clasificación de muestras desconocidas. El *conjunto de evaluación* o *test* se utiliza en el proceso de entrenamiento para validación del clasificador. Una vez entrenado el sistema, este debe ser evaluado para determinar su grado de precisión o certeza. Para ello, se proporcionan como muestras a clasificar todas las pertenecientes al conjunto de evaluación y se compara la etiqueta de salida del clasificador con su etiqueta real bien conocida.

Es común disponer de pocas muestras de entrenamiento y de test por lo que existe un amplio campo de estudio de técnicas de aprovechamiento de datos para optimizar al máximo la información contenida en los mismos. Estas técnicas consisten en particionar los datos en varios subconjuntos independientes, de forma que se puedan utilizar tanto para entrenamiento como para test y así multiplicar la potencia estadística de los resultados. Las técnicas más comunes son *Hold Out, Cross-validation* y *Leave one out*.

1.5 METODOLOGÍA

Para este trabajo se ha decidido partir de los algoritmos *Phase Correlation* y SURF. Se realizará un estudio de los diferentes métodos y de las ventajas e inconvenientes que ofrecen cada uno de ellos. Finalmente se elegirá el que mejor se adapte a los requerimientos del problema.

La metodología a seguir se dividirá en dos bloques diferenciados: El desarrollo de un prototipo experimental y la implementación de una librería final.

Para el desarrollo del prototipo experimental, se evaluará la respuesta de los diferentes algoritmos frente a las dificultades que entraña el paradigma del reconocimiento de imágenes en general y la detección de publicidad en particular. Será, por lo tanto necesario una experimentación y evaluación de los algoritmos frente a las deformaciones típicas como la rotación, el escalado o la traslación. Se evaluará también el impacto del ruido sobre estos algoritmos, así como problemas de ocultación parcial o baja resolución de las imágenes. Por último se evaluará el rendimiento y el coste computacional asociado a cada método, con el objetivo de seleccionar el que mejores prestaciones ofrezca. La experimentación propuesta se realizará sobre diferentes conjuntos de datos independientes, así como sobre un *Phantom* que nos ayude a acelerar el proceso de experimentación sin perder relevancia en los resultados.

El desarrollo del prototipo experimental se realizará bajo la plataforma MATLAB®, por ser una herramienta potente y versátil que simplifica en gran medida la consecución de toda metodología propuesta.

Por último, se implementará una librería *release* en lenguaje de alto nivel, en concreto C#, cuyo destino será la integración final en el sistema de información de inteligencia mediática definido en la introducción del proyecto. Se evaluará el correcto funcionamiento de dicha implementación sobre las dificultades anteriormente presentadas, y se realizará una comparativa del rendimiento de esta implementación con respecto al prototipo experimental.

1.6 MÉTRICAS DE EVALUACIÓN

En el reconocimiento de formas, al igual que en otras disciplinas, los resultados obtenidos por un sistema pueden englobarse en los siguientes conjuntos: *verdaderos positivos, verdaderos negativos, falsos positivos* y *falsos negativos*. Estos términos se utilizan como estimadores de la calidad de un sistema [12]. Los *falsos positivos* representan aquellas detecciones o generaciones de resultados que no deberían haber ocurrido, y que por tanto que son erróneas. Por otro lado, los *falsos negativos* son otro

valor a tener en cuenta en un sistema de clasificación. Los *falsos negativos* son aquel conjunto de resultados correctos que el sistema no ha conseguido generar. En concreto, son aquellos casos en los que no se ha logrado detectar la presencia de un logotipo en una página publicitaria que sí lo contenía, y por lo tanto no se está evaluando correctamente el impacto publicitario de la marca. Para finalizar, los *verdaderos positivos* y los *verdaderos negativos* son aquellos casos que se corresponden con los resultados esperados, es decir, representan las detecciones correctas de logotipos existentes y los casos en los que no existe el logotipo y el sistema acierta al no generar ningún resultado.

Estos conjuntos de valores pueden traducirse en dos estimadores muy importantes llamados *sensibilidad* y *especificidad* anteriormente ya comentados. La *sensibilidad* es la capacidad de un sistema de detectar los verdaderos positivos. Es decir, es la probabilidad del sistema de detectar todos los logotipos existentes en la base de datos. Se define como:

$$Sensibilidad = \frac{TP}{TP + FN}$$

La *especificidad* es la capacidad de un sistema de detectar los verdaderos negativos. En otras palabras, es la probabilidad de no generar resultados erróneos. Se define como:

$$Especificidad = \frac{TN}{TN + FP}$$

Ambos conceptos están muy ligados entre sí. El objetivo ideal es conseguir un valor igual a 1 en ambos estimadores, sin embargo el incremento de uno puede llevar al decremento del otro. Por ejemplo, pongamos una base de datos sencilla de 10 páginas publicitarias en las que se encuentran 5 logotipos repartidos entre sus páginas. Una forma inmediata de obtener el valor óptimo de *sensibilidad* es asegurar que en cada una de las páginas se encuentran los 5 logotipos buscados. Con esto tenemos que: $Sensibilidad = \frac{5}{5+0} = 1$, el valor máximo. Sin embargo, si analizamos la *especificidad* observaremos que obtenemos un valor pésimo: $Especificidad = \frac{0}{0+45} = 0$, el mínimo. Para conseguir un valor óptimo de *especificidad* el razonamiento es análogo pero asumiendo que no existe ninguno de los logotipos en la base de datos. En ese caso los valores que obtendríamos serían: $Sensibilidad = \frac{0}{0+5} = 0$ y $Especificidad = \frac{5}{5+0} = 1$.

En definitiva, la *sensibilidad* y la *especificidad* nos ofrecen unos estimadores de la calidad y la precisión de nuestro sistema. En el ejemplo propuesto, el sistema es completamente inútil puesto que no realiza ninguna detección real. Ambos estimadores así lo reflejan puesto que cuando se obtiene un valor elevado en uno de ellos, el otro desciende en picado alcanzando valores mínimos. La *sensibilidad* y la *especificidad*, como sus nombres indican, proporcionan una medida de lo sensible y preciso que es el sistema.

1.7 RESULTADOS

Los resultados obtenidos por el sistema de reconocimiento de logotipos y marcas han cubierto en su totalidad, y en ocasiones mejorado, todos los requisitos funcionales especificados en la definición del proyecto. Se ha logrado diseñar un sistema capaz de localizar en grandes bases de datos, conjuntos de logotipos definidos por el usuario con un margen de error mínimo. Sobre las bases de datos evaluadas, se ha conseguido alcanzar tasas de acierto en torno al 93%, con una tasa de falsos positivos del 0.01%. En cuanto a los estimadores *sensibilidad* y *especificidad*, se han obtenido valores cercanos al 93% y 99.5% respectivamente. El sistema a su vez ofrece un muy buen rendimiento en cuanto a coste computacional, obteniendo tiempos de procesamiento y extracción de características de en torno a 1

segundo por logotipo y 17 segundos por página publicitaria, y tiempo de *matching* entre ambas de 1.4 segundos. También se ha conseguido diseñar una arquitectura generalista cuya utilidad no se limite únicamente al ámbito publicitario, sino que se pueda migrar fácilmente a otras áreas tanto científicas como cotidianas. Por parte de *Auditmedia* y del proyecto REIMED, se han cubierto todos los requisitos funcionales especificados en la definición del proyecto, y actualmente se encuentra una implementación del sistema en producción, para satisfacer la demanda de los clientes de la empresa.

Por otro lado, los esfuerzos se centran ahora en la optimización del sistema. En el proyecto se han propuesto algunas optimizaciones con respecto a coste computacional y a calidad de resultados. Algunas de las líneas futuras consisten en continuar desarrollando estas optimizaciones para mejorar el sistema y centrarse en el pre-proceso de las imágenes para mejorar la calidad de las mismas de cara al algoritmo de extracción de características. Sin embargo, el objetivo futuro más importante y ambicioso consiste en trasladar esta tecnología al campo médico. Es mi intención evaluar y ajustar dicha herramienta con el fin de especializar el sistema y orientarlo a un uso sobre el ámbito clínico.

2 MATERIALES

2.1 BASES DE DATOS DE PÁGINAS PUBLICITARIAS

Contamos con una amplia base de datos de imágenes de periódicos y revistas para el diseño del sistema. Por parte de la empresa *AuditMedia*, es posible adquirir cuanta información referida a imágenes publicitarias sea necesaria. Disponemos pues de meses completos de periódicos como *La Razón, ABC, El Mundo, El País,* etc., así como de diferentes revistas publicitarias como *Vogue, Cuore, Lonley Plante, Lecturas, Gentleman,* etc. Es obvio que, debido a esta gran cantidad de información, no se dispone de todas las muestras etiquetadas. Así pues, se construyeron tres bases de datos a partir de la información etiquetada que se disponía.

2.1.1 BASE DE DATOS MONO-MEDIO

En primer lugar hemos diseñado una base de datos sencilla de un solo medio para aproximar el problema al caso más simple. Hemos trabajamos sobre el periódico *La Razón* por ser el medio del cual disponíamos de más información etiquetada. La base de datos *Mono-Medio* consta de 64 imágenes publicitarias de un solo medio de comunicación, *La Razón*, entre las que se encuentran un total de 27 positivos. Los 27 positivos se reparten en 20 páginas diferentes de la base de datos, por lo que existen 44 páginas en las que no se encuentra ningún logotipo.

2.1.2 BASE DE DATOS MULTI-MEDIO

Puesto que la base de datos *Mono-Medio* está únicamente formada por páginas publicitarias de un solo medio, ha sido necesario construir otra base de datos de carácter más generalista, con el fin de evaluar de una forma más precisa el rendimiento del sistema. Puesto que la base de datos *Mono-Medio* está formada únicamente por un solo medio, es obvio que también representa solamente un tipo de medio, los periódicos. Es necesario pues, construir una base de datos más general que contemple diferentes medios de comunicación, y también diferentes tipos de medios, como por ejemplo periódicos, revistas, semanales, etc.

Así pues, se ha diseñado la base de datos *Multi-Medio,* que está formada por 40 páginas aleatorias de los medios: *La Razón, Lonely Planet, Vogue* y *ABC*. En total se encuentran en la base de datos 53 positivos repartidos en las 40 páginas de la base de datos. Por lo tanto, en este caso no existe ninguna página publicitaria que no contenga como mínimo algún logotipo.

2.1.3 BASE DE DATOS TEST INDEPENDIENTE

Las bases de datos *Mono-Medio* y *Multi-Medio* han sido utilizadas durante el proceso de análisis, entrenamiento y experimentación del sistema. Por este motivo, ha sido necesario construir una base de datos independiente, de muestras desconocidas para el sistema, para evaluar y validar los modelos de clasificación y el sistema en general.

La base de datos *Test Independiente* está formada por páginas publicitarias provenientes del periódico *El Mundo* y las revistas *Cuore y Gentleman*. La base de datos consta de 50 páginas publicitarias en las que se encuentra un total de 45 positivos repartidos en 35 páginas diferentes y por lo tanto, restan 15 páginas en las que no se encuentra ningún logotipo.

2.2 BASES DE DATOS DE LOGOTIPOS Y MARCAS

Las bases de datos anteriormente comentadas están formadas por páginas extraídas de medios de comunicación escritos, dentro de los cuales se pueden encontrar diferentes logotipos. El objetivo del sistema que se pretende diseñar en este trabajo consiste en detectar en los medios publicitarios, la

aparición de imágenes corporativas de una determinada entidad. La detección de estos anuncios se corresponde con la localización del logotipo de la entidad sobre las diferentes páginas del medio, sin embargo es frecuente que una misma entidad disponga de varios logotipos. En este punto es necesario distinguir entre los conceptos de *logotipo* y el concepto *marca publicitaria*. Una *marca publicitaria* se puede entender como un contenedor de varios logotipos que pueden ser completamente diferentes pero que representan la misma entidad. Un ejemplo sencillo es el mostrado en la Figura 2.1. En esta figura se aprecia como es necesario diferenciar entre *logotipos* y *marcas publicitarias.*

FIGURA 2.1: DIFERENTES LOGOTIPOS DE UNA MISMA MARCA. RENFE

Es necesario englobar este conjunto de logotipos en una marca común, para realizar un cálculo preciso de la influencia de una entidad sobre un medio de comunicación. La solución a este problema consiste en construir un diccionario o estructura de datos que almacene las posibles variaciones de los logotipos para cada una de las marcas publicitarias. De esta forma, si se solicita al sistema que analice el impacto publicitario de la marca *Renfe*, será necesario evaluar cada uno de los logotipos de la Figura 2.1 frente a toda la base de datos. Esto es necesario puesto que no se tiene información a priori de cuál de ellos aparecerá, o incluso si aparecerán varios a la vez. Así pues, el uso de un diccionario de variaciones para cada marca es un elemento indispensable para cubrir las necesidades del sistema.

Sin embargo, la empresa *AuditMedia* especificó una restricción sobre este problema. Se indicó que se evaluara por separado cada logotipo, es decir, que no se construyera un diccionario que recogiera conjuntos de logotipos bajo una misma marca. De esta forma, el modelo de uso del sistema pasaba a ser el siguiente: si un cliente quiere hacer uso del sistema, tendrá un máximo de N logotipos a evaluar. Se tomará cada uno de estos logotipos independientemente y finalmente se le indicará cuantas apariciones de dicho logotipo se han encontrado en la base de datos seleccionada. Mediante esta restricción se simplifica el problema, reduciéndose a detecciones aisladas de logotipos independientes.

2.2.1 Conjunto de logotipos para la base de datos Mono-medio

Con respecto a la base de datos *Mono-Medio*, se crearon dos conjuntos de logotipos diferentes. En primer lugar se evaluó un conjunto de 26 *marcas publicitarias* con no más de 5 variaciones de logotipo por *marca*. De las 26 *marcas*, 22 aparecían en la base de datos (al menos un logotipo de la *marca*) y 4 se introdujeron intencionadamente como negativos.

Atendiendo a la restricción de considerar logotipos independientes sin agruparlos bajo una misma marca, se construyó un conjunto de 22 logotipos, formado por cada uno de los logotipos que aparecían en dicha base de datos.

2.2.2 Conjunto de logotipos para la base de datos Multi-medio

Con respecto a la base de datos *Multi-Medio,* se construyó un conjunto de logotipos formado por los 41 logotipos independientes, repartidos en las 40 páginas publicitarias de las que consta la base de datos. En dicha base de datos se encuentran en total 53 apariciones de los 41 logotipos de forma que todos los logotipos aparecen al menos una vez en y algunos se repiten varias veces.

2.2.3 CONJUNTO DE LOGOTIPOS PARA LA BASE DE DATOS TEST INDEPENDIENTE

Para la base de datos *Test Independiente* se ha construido un conjunto de logotipos formado por los 29 logotipos independientes contenidos en dicha base de datos. En total se encuentran 45 apariciones de los 29 logotipos de forma que todos los logotipos aparecen al menos una vez y algunos se repiten varias veces.

2.3 *PHANTOM*

Debido a las restricciones de complejidad de determinados experimentos y a la espera de una base de datos etiquetada más amplia, se desarrolló una alternativa que permitiera la consecución del trabajo a mayor velocidad y con un coste menor. Dada la base de datos *Multi-Medio*, un experimento consiste en la comparación de 41 logotipos frente a 40 páginas publicitarias. El coste computacional y el tiempo de espera en la obtención de los resultados, son demasiado elevados como para asumirlos en un proceso de experimentación. Para este propósito se hace necesaria una metodología ágil que permita explorar diferentes alternativas sin un alto coste computacional. Por este motivo, se ha desarrollado una estrategia potente que permita minimizar el coste experimental. La estrategia consiste en el diseño de un *Phantom* que recoja las posibles situaciones o entornos en los cuales puede presentarse un logotipo.

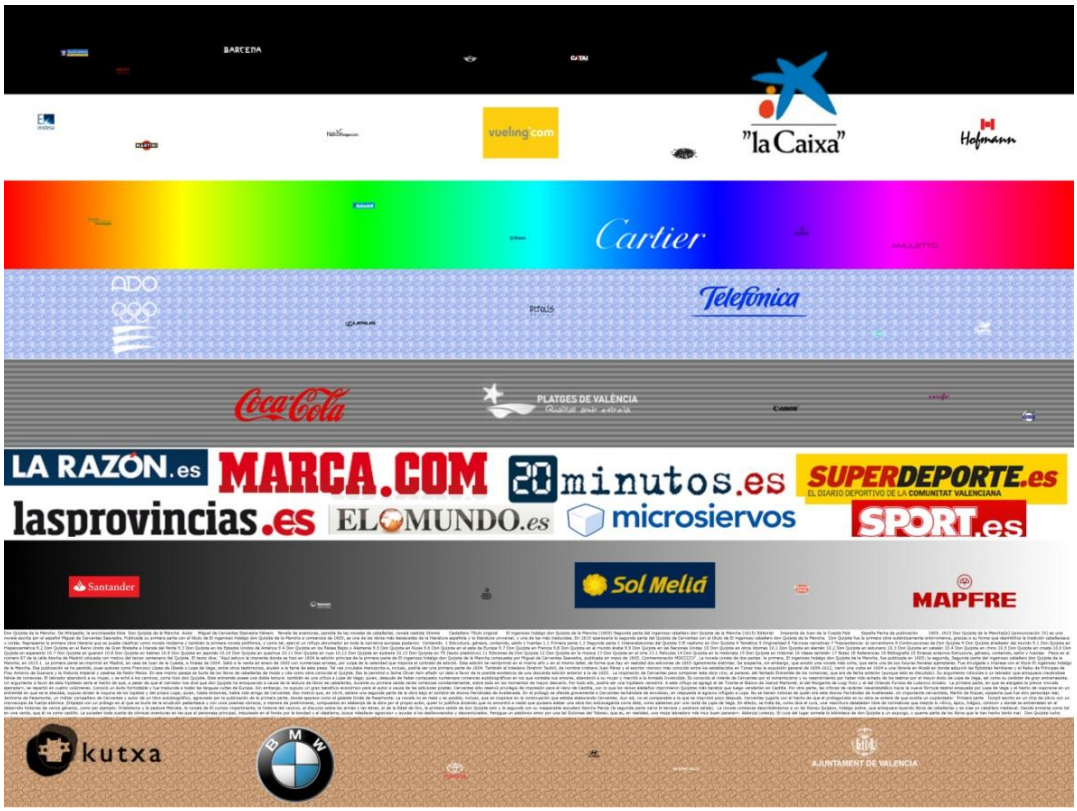

FIGURA 2.2: PHANTOM

En primer lugar, se realizó una búsqueda de *Phantoms* en la literatura, no encontrándose ningún estándar que nos sirviera como referencia para el proceso de experimentación y comparación de resultados. Por este motivo, nos ha sido necesario diseñar desde cero un *Phantom* propio que atendiera a las necesidades que se presentaban en el estudio.

El *Phantom* diseñado en este estudio es una imagen de 6000×4500, en torno a un poco más del doble que una imagen publicitaria común. En el *Phantom* encontramos dos elementos básicos que simulan la dificultad del proceso de detección: el fondo y las variaciones sobre los logotipos. Por lo que

respecta al primero, se han utilizado distintos fondos organizados por bandas donde cada una de las bandas representa una posible situación en la que se puede encontrar un logotipo. Se han diseñado bandas con colores planos o mate (bandas 1 y 2), bandas con degradados de color (bandas 3 y 7), zonas con texturas (bandas 4 y 9), áreas de cambios bruscos de gradiente (banda 5) y zonas de rótulos y texto frecuentemente encontradas en páginas de periódico o revistas (bandas 6 y 8).

FIGURA 2.3: DETALLE CAMBIO DE GRADIENTE (BANDA 5) Y TEXTURA EN EL PHANTOM (BANDA 4)

Por lo que respecta a las variaciones sobre los logotipos, se han repartido aleatoriamente por el *Phantom,* los 41 logotipos de la base de datos *Multi-Medio* a los que se le ha aplicado las transformaciones típicamente usadas en publicidad: escalados y siluetas. Por silueta entendemos la representación de todos los elementos del logotipo en un único color mate, normalmente blanco o negro.

FIGURA 2.4: EJEMPLO DE LOGOTIPO ORIGINAL Y SILUETA

Mediante el uso de este *Phantom* conseguimos reducir drásticamente el coste computacional y temporal de un experimento. Dada la base de datos *Multi-Medio*, el coste de un experimento de 40 imágenes publicitarias frente a 41 logotipos, es de $40 * 41 = 1640$ comparaciones. Mediante el *Phantom*, este proceso se reduce a un experimento de 1 imagen frente a 41 logotipos, es decir $1 * 41 = 41$. Como se puede ver, la reducción del coste es muy considerable. Sin embargo la mayor ventaja es que la experimentación no pierde relevancia estadística, es decir, la capacidad de generalización de los resultados es similar a la de los resultados obtenidos a partir de ejemplos reales. Esto es posible debido al diseño del *Phantom*, ya que se ha diseñado representando un entorno lo suficientemente generalista en el que se cubren la gran mayoría de situaciones reales a las que se puede enfrentar el sistema. Es más, uno de los objetivos del *Phantom* consiste en poner a prueba el sistema. Existen muchas situaciones límite en el *Phantom*, como la quinta banda, cuyo objetivo es elevar al máximo el grado de dificultad de la extracción de puntos y la detección posterior del logotipo. Las bandas de texturas representan también situaciones muy comunes en anuncios, en los que el fondo constituye un paraje o una determinada textura sobre la cual se sitúa el logotipo. Este tipo de fondos introducen mucha variabilidad y ruido, que debe ser tratados de forma correcta para separar la información del fondo, de la información del logotipo.

En conclusión, esta estrategia nos ha sido de gran ayuda para ajustar y evaluar el rendimiento del sistema, evitando sufrir un alto coste computacional y preservando la potencia estadística de los resultados.

3 MÉTODOS

Este capítulo recoge las diferentes técnicas y metodologías empleadas en el proyecto para el desarrollo del sistema de detección de imágenes basadas en contenido. El capítulo se divide en dos bloques generales, correspondientes a la primera aproximación tomada para abordar el problema y a la metodología final utilizada para el diseño del sistema. Este último bloque se divide a su vez en varios bloques correspondientes a cada una de las etapas que forman un sistema de reconocimiento de formas y a la presentación de la implementación final de la librería de uso comercial.

3.1 PRIMERA APROXIMACIÓN. *PHASE CORRELATION*

El algoritmo *Phase Correlation* se aplica sobre el campo de las señales y proporciona una medida de similitud entre dos señales en función de un retraso *temporal* que se aplica a una de ellas. Comúnmente, es utilizado para buscar una señal de larga duración a partir de una más corta y bien conocida. En el ámbito de las imágenes, *Phase Correlation* trata de determinar el movimiento o traslación relativa entre dos imágenes, de forma que el punto de mayor correlación se alcanza cuando las señales superpuestas de ambas imágenes no se encuentren desplazadas en ninguna fase.

Dadas dos imágenes i_1, i_2, en primer lugar se aplica un ventanado a ambas para evitar las discontinuidades en los bordes de la imagen. Debido a que *Phase Correlation* utiliza la transformada de *Fourier*, la señal a procesar se considera infinita y es necesario aplicar esta técnica para replicar la señal y evitando que se produzcan saltos bruscos. Si no fuera así, en la descomposición de *Fourier* aparecerían componentes de altas frecuencias necesarias para explicar dicha discontinuidad.

Acto seguido, se calcula la transformada discreta de *Fourier*.

$$I_1 = \mathcal{F}\{i_1\}, I_2 = \mathcal{F}\{i_2\}$$

Se utiliza una aproximación muy rápida de dicha transformada que se llama *Fast Fourier Transform (FFT)* cuyo coste computacional para una señal de *N* puntos es $\mathcal{O}(N \log N)$ mientras que el cálculo de la transformada discreta original conlleva un coste de $\mathcal{O}(N^2)$. Este algoritmo ha sido calificado en varias ocasiones como uno de los algoritmos más importantes en los últimos años.

Una vez calculada la transformada de *Fourier* se calcula el espectro cruzado de potencia (*cross-power spectrum*) con el fin de disminuir la influencia de las variaciones de la luminosidad sobre la señal. Para ello, se multiplica punto a punto la primera transformada por el complejo conjugado de la segunda, y se normaliza el producto.

$$R = \frac{I_1 I_2^*}{|I_1 I_2^*|}$$

A continuación se computa la inversa de la transformada de *Fourier* para obtener la expresión normalizada de *cross-correlation*.

$$r = \mathcal{F}^{-1}\{R\}$$

En último lugar, se determina dónde se encuentra el pico de mayor intensidad que se corresponderá con la mayor correlación encontrada sobre la imagen.

$$(\Delta x, \Delta y) = argmax_{(x,y)}\{r\}$$

A diferencia de otros algoritmos del dominio frecuencial, *Phase Correlation* es robusto al ruido, oclusión y otros defectos típicos de las imágenes. Sin embargo, este método presenta grandes inconvenientes para la tarea que se desea resolver. La representación de una imagen mediante una

señal es débil frente a muchas transformaciones típicas de las imágenes y muy comunes en publicidad. La dependencia por ejemplo de la escala o rotación de las imágenes, es un grave inconveniente que, aunque es posible extender el algoritmo de forma no natural para trabajar en estas situaciones, conlleva un alto coste computacional y provoca que el algoritmo pierda gran parte de su eficiencia y rendimiento. Para solucionar el problema del escalado, se construirá un diccionario de 200 imágenes de diferentes tamaños para cada logotipo. El rango de escalas que contendrá cada diccionario comprenderá desde el tamaño máximo que pueda alcanzar cada logotipo, sin deformar su *aspect ratio*, hasta 10 veces menos el tamaño original de los logotipos. El algoritmo tomará cada imagen del diccionario y aplicará el método *Phase Correlation* en busca del mayor grado de correlación.

3.2 Diseño final. Speeded-Up Robust Features

El diseño final del sistema de reconocimiento de logotipos y marcas se ha realizado basándose en el algoritmo de extracción de características en imágenes *Speeded-Up Robust Features (SURF)*. A continuación se presenta la metodología seguida para el diseño del sistema, dividida en 4 sub-apartados correspondientes a las etapas de un sistema de reconocimiento de formas: Pre-proceso, extracción de características, clasificación o *matching* y post-proceso, y un apartado final en el que se presenta la implementación de la librería final de uso comercial en lenguaje de alto nivel.

3.2.1 Pre-Proceso

El Pre-Proceso es la primera etapa dentro de un sistema de reconocimiento de formas. El objetivo principal de esta etapa consiste en la mejora de la información contenida en la imagen, eliminando distorsiones no deseadas y/o realzando algunas características importantes de la misma para su posterior procesamiento o análisis.

En este estudio se han considerado los siguientes métodos de pre-proceso: Inversión de color, filtrados, PCA, y el ajuste del lienzo de la imagen.

3.2.1.1 Técnicas sobre píxel

3.2.1.1.1 Inversión de color

En el ámbito publicitario es común encontrarse logotipos a los cuales se les ha aplicado algún tipo de transformación típica. Estas transformaciones suelen ser escalados, rotaciones y siluetas. El método de extracción de características *SURF*, es invariante a las dos primeras, sin embargo es sensible a la última. Por silueta entendemos la representación del logotipo en un color uniforme, típicamente blanco o negro, y fijo para todos los elementos del mismo. Es necesario entonces diseñar un método que ofrezca buenos resultados ante este tipo de transformaciones.

El principal problema de la silueta es la diferencia del tipo de contraste entre los elementos del logotipo original y los de la propia silueta. Si tomamos el ejemplo de la Figura 3.1 y pensamos en ambas imágenes en escala de grises, el logotipo original presenta un tipo de contraste de elementos grises sobre un fondo blanco, mientras que en la imagen de la silueta, el contraste es el contrario. Si extraemos puntos interesantes de ambas imágenes y los comparamos, estos no se asemejaran debido a esta diferencia de contrastes. Es por tanto necesario abordar estas situaciones debido a su frecuente aparición en los medios. La aproximación más sencilla para solucionar el problema y que mejor resultados ofrece es la inversión de color del logotipo.

FIGURA 3.1: LOGOTIPO *AJUNTAMENT DE VALENCIA* Y EJEMPLO DE SU SILUETA

El método es extremadamente sencillo. La única operación necesaria consiste en restar el valor máximo que puede tomar un píxel, a cada uno de los píxeles de la imagen. El valor máximo para imágenes con profundidad de color de 8 bits es 255. Formalizando la expresión tenemos:

$$para\ i = 1\ hasta\ Ancho\ imagen$$
$$para\ j = 1\ hasta\ Alto\ imagen$$
$$ImagenInvertida(i,j) = 255 - Imagen(i,j)$$

Una vez obtenida la imagen invertida, es necesario extraer los puntos interesantes de la misma y almacenarlos para su posterior comprobación. De esta forma un logotipo se define mediante dos vectores de puntos interesantes correspondientes a los puntos extraídos de la imagen original y los puntos extraídos de la imagen invertida en color.

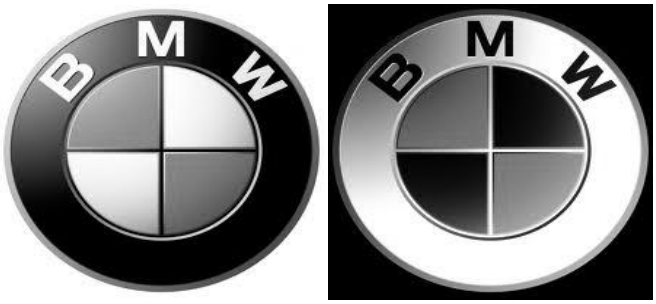

FIGURA 3.2: LOGOTIPO ORIGINAL E INVERTIDO EN ESCALA DE GRISES

En el posterior proceso de *matching,* será necesario evaluar ambos conjuntos de puntos contra todos los puntos extraídos de la página publicitaria sobre la que se desea localizar el logotipo. Así pues, dado un conjunto de N logotipos y un conjunto de M imágenes publicitarias, el coste computacional de la extracción de puntos es $2\alpha N + \beta M$, siendo $N \ll M \ \wedge\ \alpha, \beta$ coeficientes del coste de extracción de puntos para los logotipos y las páginas publicitarias. Los valores aproximados que pueden tomar estos coeficientes son $\alpha = 0.05\ y\ \beta = 0.95$ correspondientes a los pesos computacionales que tienen en el proceso de extracción de características. Por otro lado, además de que generalmente N es bastante menor que M, estas imágenes a su vez son también bastante inferiores en tamaño. Normalmente el tamaño de una página publicitaria se encuentra entre 2300×3000 píxeles mientras que un logotipo no suele superar los 500×500. Esta característica permite aplicar pre-procesos a los logotipos que, aunque impliquen varios cálculos de puntos interesantes, no influyan notablemente en el coste computacional general del sistema. Sin embargo, aplicar operaciones sobre las páginas publicitarias implica un mayor coste computacional, y por tanto conviene evitarlas.

3.2.1.2 FILTRADO

3.2.1.2.1 FILTRO DE REALCE DE BORDES. *CANNY EDGE DETECTOR*

El objetivo principal de la detección de bordes, es capturar cambios importantes de intensidad y brillo en las imágenes, que representen propiedades *físicas* del mundo real. Puede demostrarse, bajo asunciones generales de los modelos de formación de las imágenes, que estos cambios o discontinuidades, se corresponden con alguna de las siguientes características:

- Discontinuidades en la profundidad.
- Discontinuidades en la superficie u orientación.
- Cambios en las propiedades de los materiales.
- Variaciones en la iluminación de las escenas.

Así pues, los *bordes* en una imagen se definen como los píxeles cuya función de intensidad cambia de forma repentina o brusca. Los *filtros de realce de bordes* o *detectores de bordes* son colecciones de métodos de pre-proceso de imágenes cuyo objetivo es encontrar aquellas regiones de la imagen cuyo cambio de intensidad o brillo es muy pronunciado. Este tipo de filtros, expresan dichos cambios mediante derivadas parciales, por lo que un cambio en la función de intensidad de una imagen puede describirse por un gradiente cuya dirección es la de mayor crecimiento de la función.

Existen varios detectores de bordes como por ejemplo el detector *Canny*, el *Sobel,* el propuesto por *Prewitt* o el detector *Roberts cross*. Para este estudio, se ha utilizado el detector *Canny* por ser actualmente el que mejor prestaciones presenta.

John F. Canny desarrolló en 1986 un algoritmo de detección de bordes sobre imágenes cuyo nombre se conoce actualmente como *Canny detector* [13]. A su vez, desarrolló una teoría computacional acerca de la detección de bordes que se explicaba a través de su propio método. El objetivo de *Canny* era la búsqueda del algoritmo óptimo de detección de bordes que cumpliera determinadas condiciones como optimalidad, buena localización y alto rendimiento. Su estudio derivó en el diseño del *operador Canny* que es actualmente uno de los algoritmos más potentes de detección de bordes.

El filtro de *Canny* localiza los bordes mediante la búsqueda de máximos locales del gradiente de la imagen. Este gradiente se calcula usando un filtro basado en la primera derivada de una gaussiana y consta de dos *thresholds* o umbrales para detectar tanto los bordes pronunciados como los bordes más suaves. El método solo incluye en el resultado final los bordes suaves, si están conectados a un borde pronunciado. Por último, está demostrado [14] que el filtro *Canny* es más robusto al ruido con respecto a otros filtros de detección de bordes y por lo tanto ofrece una precisión y rendimiento mayor.

3.2.1.2.2 FILTRO DENOISING

Los filtros de eliminación de ruido son una herramienta muy útil en el pre-proceso de las imágenes o señales. En la mayoría de casos estos filtros tratan de eliminar el posible ruido contenido en una imagen filtrando principalmente las frecuencias altas y dejando pasar las bajas. Los filtros de eliminación de ruido basan su funcionamiento en la redundancia que presentan las imágenes en determinadas regiones de la misma. Así pues, dado un píxel defectuoso, es posible reemplazar dicho píxel por otro construido a partir de la media de los píxeles cercanos. Existen diferentes tipos de ruidos a los que se puede someter una imagen. Principalmente estos ruidos son los denominados ruidos *sal y pimienta (salt and pepper)*, ruido gaussiano o de amplificación, *shot noise*, ruido de granulación, etc., para los que existen varios tipos de filtros, cada uno de ellos enfocado a un determinado tipo de ruido.

En el caso que se estudia en este trabajo, las imágenes digitales, no presentan ningún ruido de los anteriormente mencionados sino un defecto derivado de la compresión JPEG, que se tratará de minimizar mediante esta técnica. Como se ha explicado en el capítulo 1.4.2, las imágenes digitales se categorizan en dos grandes tipos dependiendo de si la resolución es fija o dinámica. Estos tipos son: *imágenes rasterizadas* si la resolución es fija, e *imágenes vectoriales* si la resolución es dinámica (para más detalle revisar el capítulo 1.4.2). Las *imágenes rasterizadas* suelen comprimirse finalmente para ser almacenadas en un fichero, por lo que en este proceso se introducen pérdidas de información normalmente asociadas con la calidad de la imagen. El material con el que cuenta el sistema son *imágenes rasterizadas* comprimidas con el algoritmo JPEG. Esta combinación es muy corriente ya que consigue una alto ratio de compresión sin una perdida excesiva de información con respecto al ojo humano, sin embargo de cara a una máquina y en concreto a algoritmos de extracción de características, se convierte en un tándem no deseable.

El algoritmo de compresión JPEG es también famoso por introducir durante el proceso de compresión unos defectos llamados *artefactos JPEG*, que introducen una alta cantidad de información no deseada y cuyas características son similares al ruido. Estos artefactos dificultan mucho la tarea de reconocimiento puesto que pueden distorsionan los descriptores que se obtengan de la imagen. El objetivo pues del filtro *denoising* es eliminar estos artefactos aproximándolo como si de un ruido gaussiano se tratara.

Se utilizará el filtro *Wiener* que se desarrolló en 1940 por *Norbert Wiener* y se publicó finalmente en 1949 [15]. El equivalente en tiempo discreto del filtro de *Wiener* fue el propuesto por *Kolmogorov* en 1941 de forma independiente [16], por lo que la teoría del funcionamiento del filtro se llama: teoría de *Wiener-Kolmogorov*. El filtro *Wiener* asume que se dispone de información acerca de las propiedades espectrales de la señal y del ruido contenido en la misma, y establece como criterio de optimización la minimización del error cuadrático medio (*MMSE: Minimum Mean-Square Error*). Sobre la plataforma MATLAB®, el filtro *Wiener* toma como parámetros el área de influencia de los píxeles cercanos al píxel que se está procesando, y estima la media local y la desviación estándar para recalcular el valor del píxel afectado. Además toma como parámetro opcional el tipo de ruido al que se ha sometido la imagen y si este no se especifica se asume que es ruido gaussiano.

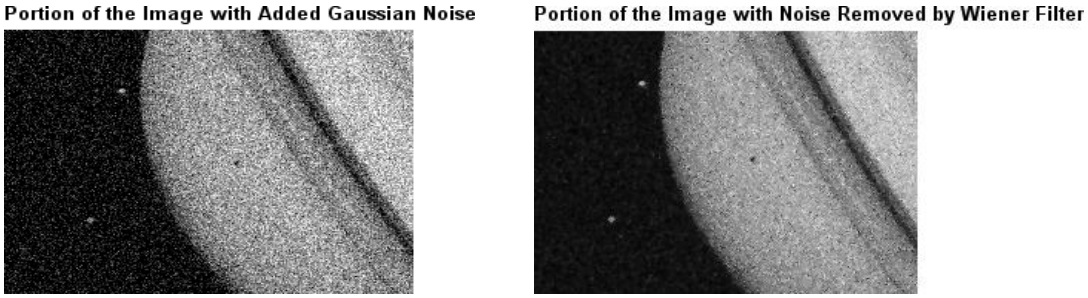

FIGURA 3.3: EJEMPLO SINTÉTICO FILTRO WIENER

3.2.1.3 GESTIÓN DE LA INFORMACIÓN DE COLOR

3.2.1.3.1 CONVERSIÓN A ESCALA DE GRISES

Las imágenes en escala de grises son imágenes en las que el valor de cada píxel representa únicamente información acerca de la intensidad de brillo. La información de color de los píxeles de la imagen se reduce a un conjunto de tonos grises que varían gradualmente desde el negro más oscuro hasta el blanco más claro.

La intensidad de cada píxel se expresa en un rango definido por un valor mínimo y máximo. Este rango se suele representar desde el 0, siendo este el valor mínimo o ausencia de brillo, es decir negro, hasta el valor 1 que indica la total presencia de brillo, es decir blanco, con todos los posibles valores fraccionados en este intervalo. Existen también otras representaciones como por ejemplo los porcentajes, siendo 0% el valor negro y 100% el blanco, sin embargo la representación habitual en informática suele ser mediante una codificación de N bits por píxel, de forma que el conjunto de valores posibles que puede tomar un píxel son 2^N niveles de gris. De la misma forma que en las anteriores representaciones, el valor 0 se corresponde con el negro más oscuro mientras que el valor $2^N - 1$ se corresponde con el blanco.

El algoritmo de conversión de una imagen en color a escala de grises es un proceso ponderado, en el que la representación final de la imagen se construye en función de las intensidades de color en cada una de los canales primarios *RGB*. Dicha conversión se pondera con unos pesos para cada canal, de acuerdo a los estudios de los expertos de la importancia que da la percepción del ojo humano a cada gama de color. Para convertir una imagen en color a escala de grises, es necesario en primer lugar obtener para cada píxel su valor de brillo en el canal rojo, verde y azul. Una vez obtenidos estos valores, el resultado final del valor de ese píxel en la imagen en escala de grises se corresponde con la suma ponderada de estas tres intensidades, añadiendo un 30% del valor del canal rojo, un 59% del valor del canal verde y un 11% del canal azul. Formalizando este proceso, la conversión a escala de grises queda como sigue:

$$para\ i = 1\ hasta\ Ancho\ Imagen$$
$$para\ j = 1\ hasta\ Alto\ Imagen$$
$$Imagen_{GrayScale}(i,j) = 0.3 \times Imagen(i,j,R) + 0.59 \times Imagen(i,j,G) + 0.11 \times Image(i,j,B)$$

donde $Imagen(i,j,X)$ representa la intensidad del píxel en la posición (i,j) de la imagen, en el canal $X \in \{R,G,B\}$.

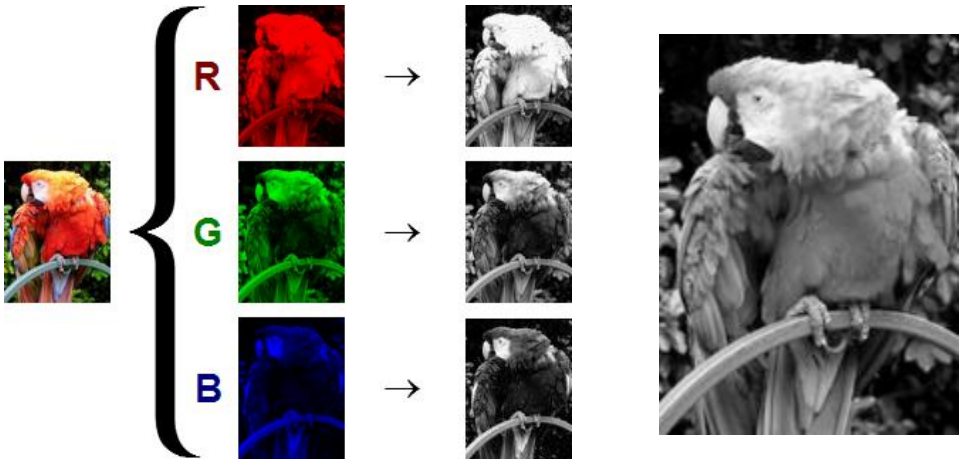

FIGURA 3.4: CONVERSIÓN DE IMAGEN EN COLOR A ESCALA DE GRISES

La conversión a escala de grises ofrece un método de reducción y simplificación de la información de color de una imagen, de forma que se facilita el tratamiento y procesamiento de las mismas con respecto a la imagen original en color.

3.2.1.3.2 Principal Component Analysis

Principal Component Analysis (PCA) es probablemente una de las herramientas más populares en el análisis estadístico multivariante. Esta técnica fue inventada en 1901 por *Karl Pearson* [17] y hoy

en día es ampliamente utilizada en muchas disciplinas para análisis de conjuntos de datos (*Data sets*) o para construcción de modelos predictivos.

PCA es una técnica matemática cuyo objetivo es la reducción de la dimensionalidad de un conjunto de datos preservando la mayor información posible. PCA utiliza una transformación ortogonal para convertir un conjunto de datos posiblemente correlacionados, en otro conjunto de variables no relacionadas llamadas *componentes principales*. Así pues, PCA busca una combinación lineal de los datos de forma que su proyección sobre el subespacio de menor dimensión generado, maximice la varianza de los datos proyectados. En otras palabras, PCA busca una combinación lineal de forma que se minimice la suma de los errores cuadráticos de los datos originales con respecto a los datos proyectados. Así pues, el número de *componentes principales* calculado por PCA es siempre menor o igual que el número de variables contenido en los datos originales.

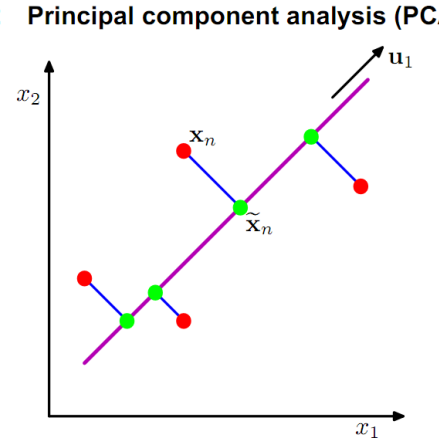

FIGURA 3.5: EJEMPLO PRINCIPAL COMPONENT ANALYSIS

PCA calcula las *componentes principales* de forma que sobre la primera componente se retenga la mayor varianza posible y en las sucesivas componentes, se retenga de igual forma la mayor varianza posible, pero con la restricción de que dicha componente debe ser ortogonal a las anteriores calculadas. En conclusión, PCA posibilita el proyectar en un nuevo sistema de coordenadas de menor dimensión el conjunto original de los datos, de forma que su representación espacial en este sistema explique la mayor variabilidad de los datos.

Por último, PCA es un método que trabaja con datos no etiquetados. El principal inconveniente de este algoritmo es que, al no considerar las etiquetas de los datos, no tiene en cuenta la existencia de distribuciones independientes o linealmente separables. Esta consideración puede derivar en que la proyección de los datos sobre el nuevo subespacio generada por PCA, mezcle o superponga ambos conjuntos de forma que estos dejen de ser linealmente separables. Este es un inconveniente de gran importancia a asumir en el uso de PCA. Es posible, como se muestra en la Figura 3.6, que exista una proyección de los datos de mejores características si tenemos en cuenta la existencia de dos distribuciones independientes. *LDA* (*Linear Discriminant Analysis*), que se expondrá en el capítulo 3.2.3, resuelve este problema como se muestra en la parte derecha de la Figura 3.6.

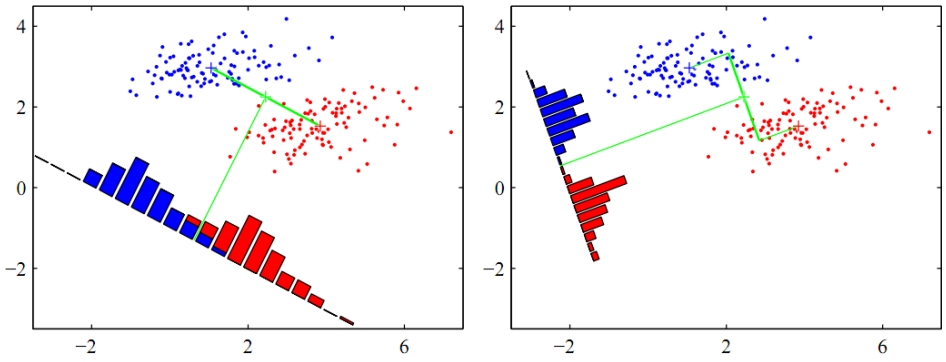

FIGURA 3.6: SUPERPOSICIÓN DE DATOS. DERECHA PCA. IZQUIERDA LDA

Una de las posibles aplicaciones de PCA en imágenes consiste en extraer los colores que contengan más información de una imagen. Dada una imagen en color definida en RGB, PCA nos proporcionaría los colores principales de la misma por orden de relevancia.

El pre-proceso consiste en aplicar PCA sobre el logotipo tomando como variables los canales RGB, y como observaciones todos los píxeles de la imagen. Una vez obtenido los colores predominantes, se proyectan todas las páginas publicitarias en las que se desee buscar el logotipo, con el fin de incrementar el contraste del mismo si es que realmente existe en alguna de las páginas. Así pues, PCA recibe una matriz de $N \times 3$, donde $N \equiv alto \times ancho$ de la imagen, y 3 las columnas correspondientes al canal rojo, verde y azul. El valor de cada uno de los elementos de la matriz es la intensidad de rojo, verde o azul dependiendo de en qué columna se observe.

El método, sin embargo, es susceptible a elementos que deben ser filtrados como por ejemplo el fondo. Es necesario eliminar el fondo en aquellos casos en los que este no forme parte del logotipo, puesto que sino distorsionaría el color principal u ocuparía una de las componentes principales del análisis. Es fácil pensar que, dada una imagen de un cuadrado rojo sobre fondo blanco, si representamos los pixeles de la imagen en el espacio de color, obtendremos dos distribuciones de puntos, una centrada en el color rojo y otra en el blanco. Al aplicar PCA sobre este conjunto de datos, el primer color principal que obtendríamos se correspondería con aquel cuya concentración de píxeles fuera mayor, probablemente el fondo. En la Figura 3.7 se propone un ejemplo real. Se presenta la distribución de puntos sobre el espacio de color de un logotipo real. Como se puede ver, existen dos distribuciones de puntos diferenciadas. Una de ellas se sitúa sobre el plano XY, centrada en $X \approx 20, Y \approx 140$ que se corresponde con los píxeles referentes a las letras verdes de “Viajes El Corte Ingles”, y otra distribución centrada en $X = 255,\ Y = 255,\ Z = 255$ que se corresponde a los píxeles blancos del fondo de la imagen. Se observa también una cola que conecta ambas distribuciones. Esta cola se debe a los píxeles del contorno de las letras que realizan el fundido del verde al fondo blanco.

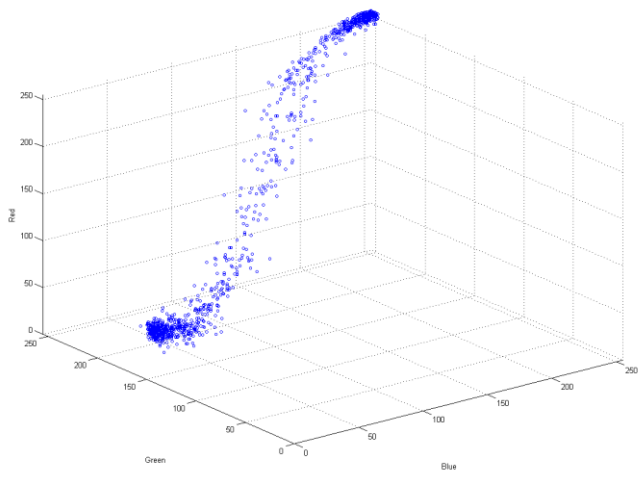

FIGURA 3.7: DISTRIBUCIÓN DE COLORES DE UN LOGOTIPO EJEMPLO DE LA BASE DE DATOS MULTI-MEDIO

Esta distribución correspondiente a los píxeles del fondo puede suponer un problema a la hora de calcular el color o colores principales de la imagen. Para solucionar este problema, se diseñó una sencilla regla que eliminara el fondo. Se parte de la premisa de eliminar únicamente fondos blancos o negros. Si un logotipo presenta un fondo de color diferente, se considera que el fondo forma parte del logotipo y por lo tanto no debe ser eliminado. En primer lugar se toman los valores de los píxeles de las cuatro esquinas de la imagen y se obtiene su media. Si esta media se encuentra por debajo de un umbral que define el color negro, o por encima de un umbral que define el color blanco, entonces se eliminan aquellos píxeles de la imagen cuyo color cumpla dicha condición, en otro caso se asume que el fondo no es ni blanco ni negro y por lo tanto forma parte del propio logotipo.

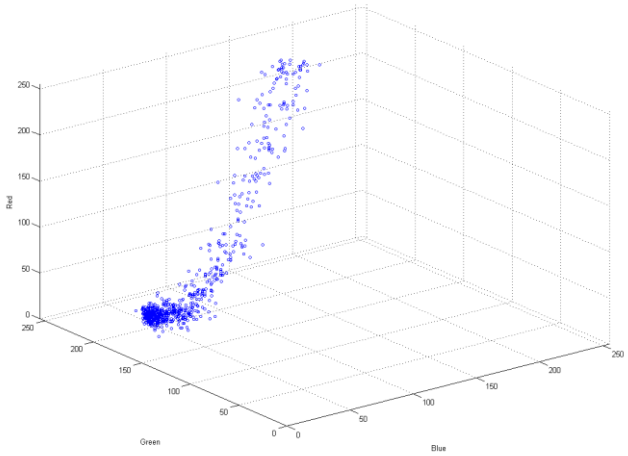

FIGURA 3.8: FILTRADO DEL FONDO DEL LOGOTIPO EJEMPLO DE LA BASE DE DATOS MULTI-MEDIO

Los umbrales que se han establecido son los siguientes: si la media de los valores de los píxeles de las esquinas supera 220 (siendo 255 el blanco puro), entonces se considera que el fondo de la imagen es blanco y se eliminan todos aquellos pixeles cuya suma de los valores de los tres canales (RGB) supere 660, es decir $220 * 3$. Para el caso del negro, el razonamiento es análogo. Si la media de los valores de los píxeles de las esquinas está por debajo de 40, entonces se considera que el fondo es negro y por lo tanto se eliminan todos aquellos píxeles de la imagen cuya suma de los tres canales de color sea inferior a $40 * 3 = 120$. En la Figura 3.8 se observa como eliminamos el fondo blanco del logotipo *Viajes El Corte Ingles* y nos quedamos únicamente con los puntos que definen el color

predominante. Es inevitable mantener cierta cola de puntos que se acercan hacia el antiguo color del fondo, sin embargo la mayor concentración de puntos se encuentra sobre el color principal del logotipo.

Para calcular PCA, sustraemos la media de la distribución y centramos los datos en el origen de coordenadas para calcular las direcciones de mayor varianza. En este supuesto, si eliminamos la media de la distribución perdemos directamente la información del color predominante, es decir, si centramos la distribución de puntos en 0 estamos transformando el color predominante en negro. Lo que buscamos por lo tanto son los vectores media de las diferentes distribuciones, correspondientes a los colores predominantes de la imagen. Para ello en este estudio aplicaremos PCA sin eliminar la media de la distribución. Una vez calculados los colores predominantes del logotipo queda proyectar las imágenes publicitarias en estos espacios de color y observar los resultados.

3.2.1.3.3 RGB DECOMPOSITION

RGB Decomposition se puede considerar una simplificación del algoritmo anterior. En PCA rotamos los ejes de colores para obtener un nuevo sistema de coordenadas propio de cada logotipo, y proyectamos todas las imágenes sobre estos ejes con el fin de resaltar dicho logotipo si es que existe realmente. En *RGB Decomposition*, se toman los ejes definidos por los colores rojo, verde y azul tanto para los logotipos como para las imágenes y se calculan los puntos interesantes en cada canal.

FIGURA 3.9: LOGOTIPO RENFE

Por ejemplo, dado el logotipo de la Figura 3.9, PCA recogería en la primera componente la información referente a los puntos de color morado de la imagen. Sin embargo este color morado, en el espacio RGB, es una combinación de rojo y azul. En *RGB Decomposition* se repartirá la información de estos puntos entre las capas roja y azul. En el caso en el que la información de color se reparta de forma equivalente entre los tres canales, el resultado obtenido será el mismo que si de una imagen en escala de grises se tratara.

El método es sencillo, se separan las capas de color de las imágenes y se extraen los puntos interesantes en cada una de ellas. Una vez extraídos dichos puntos, se comparan por pares las capas R, G y B de los logotipos con las de las páginas publicitarias. Dicho de otra forma, se compara el canal rojo del logotipo frente al canal rojo de la página publicitaria, el canal verde con su homólogo verde y de igual forma con el canal azul. Finalmente se suman los puntos o se ponderan teniendo en cuenta que se ha analizado una misma página tres veces.

Supongamos una situación como la de la Figura 3.10, en la que tenemos una imagen cuyo fondo es de color granate y sobre ese fondo se sitúa un cuadrado de color azul marino. Si transformamos esta imagen a escala de grises obtenemos la segunda representación de la Figura 3.10. Si tratamos de extraer puntos interesantes sobre esta representación es muy probable que el resultado sea nulo. SURF no observará ningún contraste entre el cuadrado y el fondo, por lo que no extraerá ningún punto interesante de la imagen. Sin embargo, si aplicamos la técnica *RGB Decomposition*, el resultado será completamente diferente. Si descomponemos la imagen en los canales rojo, azul y verde, tenemos tres representaciones siguientes de la Figura 3.10.

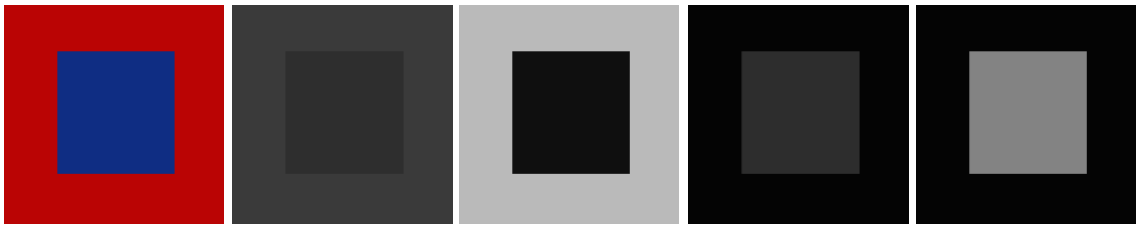

FIGURA 3.10: DE IZQUIERDA A DERECHA: IMÁGEN ORIGINAL, ESCALA DE GRISES, CANAL R, CANAL G, CANAL B

En la tercera representación se observa la información del canal rojo. Como se puede ver, el contraste es altísimo y no existiría ningún problema para discernir los elementos de la imagen y extraer los puntos interesantes correspondientes. La cuarta imagen se corresponde con el canal verde, que por no tener influencia en la imagen aparece prácticamente en negro. Por último se muestra el canal azul, en el que ocurre lo mismo que en el canal rojo pero con el contraste invertido. Mediante esta técnica, *SURF* sería capaz de extraer puntos interesantes en los canales rojo y azul y por lo tanto describir perfectamente la imagen.

3.2.1.4 AJUSTE DEL TAMAÑO DEL LIENZO

La necesidad del ajuste del tamaño del lienzo deriva de los requisitos necesarios que deben cumplir las imágenes para su correcta descripción mediante el algoritmo SURF. El algoritmo de extracción de características SURF describe cada punto interesante de la imagen en base a una cantidad de información espacial que toma del entorno, dependiendo de la escala a la que se haya detectado el punto. Es por lo tanto necesario que en cualquier punto o zona de la imagen susceptible de ser descrita, exista el suficiente espacio alrededor del mismo que permita su correcta descripción. Así pues es necesario que una imagen se encuentre bien encuadrada en el lienzo, dejando un margen suficiente desde la finalización de la figura contenida en la imagen hasta la finalización de la propia imagen.

FIGURA 3.11: DIFERENCIA DE EXTRACCIÓN DE PUNTOS CON Y SIN AJUSTE DE LIENZO

En la Figura 3.11 se muestran las grandes diferencias en cuanto a extracción de puntos de ambas imágenes. En concreto para este ejemplo, con un buen ajuste del tamaño del lienzo se han extraído 68 puntos interesantes. Por el contrario, en la imagen de mal ajuste únicamente ha sido posible extraer 30 puntos, menos de la mitad con respecto al logotipo bien ajustado.

En estos momentos no se dispone de un método cerrado para dar solución a dicho problema. Puesto que es un requisito detectado recientemente, hasta ahora se ajusta el tamaño del lienzo de forma manual. Sin embargo se están estudiando diferentes algoritmos para realizar este ajuste de forma automática. La aproximación más simple puede consistir en realizar un barrido de la imagen en anchura y altura, para detectar el primer píxel de diferente color (con un cierto umbral de variabilidad) con

respecto al color del lienzo de la imagen. Una vez detectado este píxel, se puede obtener fácilmente el margen que guarda el logotipo con el lienzo, y ajustar el mismo por ejemplo a un 25% del tamaño total de la imagen.

3.2.2 EXTRACCIÓN DE CARACTERÍSTICAS

Speeded-Up Robust Features (SURF) es un método de extracción de características para imágenes, presentado por primera vez el 7 de Mayo de 2006 en el *ECCV (European Conference on Computer Vision)* en *Graz, Austria* [8]. Este método fue creado por *Herbert Bay, Andreas Ess, Tinne Tuytelaars y Luc Van Gool* y está parcialmente inspirado en el algoritmo *SIFT (Scale Invariant Feature Transform)* [18]. *SURF* es un método invariante a escala, rotación y traslación, y constituye una herramienta de descripción y representación de imágenes. *SURF* es un método de extracción de características basado en la detección de puntos interesantes de las imágenes. Estos puntos se corresponden principalmente con bordes, *manchas*, zonas de elevado contraste, bifurcaciones, etc. Una vez detectados estos puntos, el algoritmo los describe mediante información espacial recogida del entorno del punto.

En el siguiente capítulo se explica el algoritmo en profundidad basándose en el *paper* de presentación de los autores de SURF y en la propuesta de implementación presentada en el artículo [19] que se corresponde con la utilizada en el sistema de detección de logotipos y marcas diseñado.

DETECCIÓN DE PUNTOS INTERESANTES

La detección de puntos interesantes del algoritmo SURF está basada principalmente en la matriz Hessiana, y en concreto en el cálculo del determinante de la matriz Hessiana en cada punto de la imagen. Mediante este valor, es posible determinar si en dicho punto existe un cambio elevado de contraste o una región de alta variabilidad que constituya por lo tanto un punto de interés en la imagen. En los siguientes apartados se explicará el concepto de imagen integral y sus ventajas al realizar los cálculos, la definición de la matriz Hessiana y su aplicación sobre imágenes y la definición del espacio de escalas y su representación en el algoritmo.

Imagen Integral

El uso de imágenes integrales para la detección de puntos interesantes fue propuesto por *Viola* y *Jones* [20], dónde demostraron que mediante esta técnica se reducía drásticamente el coste computacional de la detección de puntos de interés. Gran parte del buen rendimiento que ofrece el algoritmo SURF es debido al uso de esta representación intermedia de las imágenes. Como se verá a continuación, el cálculo de la suma de intensidades en una región de una imagen integral es mínimo.

El cálculo de la integral de una imagen se realiza de la siguiente forma. Dada una imagen I y un punto $X = (x, y)$, la integral de la imagen en ese punto es la suma de todos los valores de los píxeles de la imagen que forman un rectángulo desde el origen hasta dicho punto X:

$$I_{\Sigma}(X) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j)$$

En la Figura 3.12 se muestra como, una vez calculada la integral de la imagen, el cálculo de la suma de intensidades de cualquier área rectangular de esa imagen se reduce únicamente a tres sumas. Esta propiedad es de gran importancia puesto que ofrece una alternativa para el cálculo de este valor de extremada rapidez. Más aun, este procedimiento ofrece una potentísima ventaja. El coste de realizar el cálculo de la suma de intensidades en cualquier región de la imagen, es independiente y siempre el

mismo para cualquier tamaño de dicha región. Es decir, el coste del cálculo de la suma de intensidades en una región de 20×20 píxeles es el mismo que el coste del cálculo de la suma de intensidades sobre una región de 2000×3000, en concreto tres sumas. Esta propiedad es de gran utilidad cuando se requiere calcular las intensidades de grandes áreas en una imagen. SURF hace un muy buen uso de esta propiedad computando grandes cálculos de filtros de diferentes tamaños, en un tiempo casi constante.

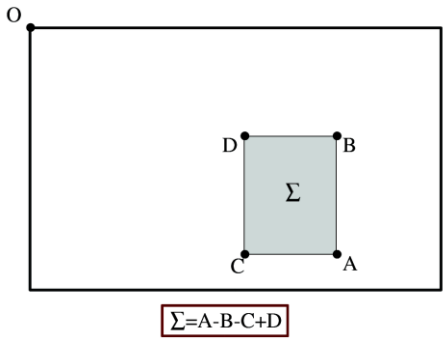

FIGURA 3.12: CÁLCULO DE LA SUMA DE INTENSIDADES EN UNA IMAGEN INTEGRAL

Matriz Hessiana

A diferencia de otros algoritmos como SIFT que basan la detección de puntos interesantes en el operador laplaciano, SURF realiza la detección de puntos de interés sobre la matriz Hessiana. En concreto, SIFT hace uso del operador *DoG (Difference of Gaussian)* que es una aproximación rápida del operador *LoG (Laplacian of Gaussian)*, mientras que SURF trabaja con el determinante de la matriz Hessiana, *DoH (Determinant of Hessian)*, ya que ofrece unas condiciones de precisión mayor. Para explicar el uso de la matriz Hessiana, consideraremos una función continua de dos variables $f(x, y)$. La matriz Hessiana H, es la matriz de las segundas derivadas parciales de la función f.

$$H\big(f(x,y)\big) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

El determinante de esta matriz se calcula de la siguiente forma:

$$\det(H) = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2$$

El valor de este determinante se puede utilizar para clasificar el máximo y el mínimo de la función f. Puesto que el determinante es el producto de los valores propios de la matriz Hessiana, es posible clasificar los puntos de la función basándose en el signo del resultado. Si el determinante es negativo entonces los valores propios son de diferente signo y por lo tanto ese punto no representa un extremo local. Por el contrario, si el valor es positivo, entonces ambos valores propios son positivos o ambos negativos y por lo tanto en ese caso se pueden clasificar como un extremo local.

Trasladar esta teoría para trabajar sobre imágenes en vez de sobre funciones continuas es trivial. En primer lugar reemplazamos los valores $f(x, y)$ por los valores de la intensidad de la imagen en cada píxel $I(x, y)$. En segundo lugar es necesario un método para calcular las derivadas parciales de segundo orden de la imagen. Estas derivadas se pueden calcular mediante una convolución con un *kernel* apropiado. El *kernel* que utiliza SURF para esta convolución es un *kernel* gaussiano de segundo orden. El algoritmo construye diferentes *kernels* en las direcciones de x, y y xy para calcular los cuatro

valores de la matriz Hessiana. El uso de este *kernel* además, es el que permite calcular el determinante de la matriz Hessiana a diferentes escalas, por lo que, llegados a este punto, podemos calcular la matriz Hessiana H como una función relativa al espacio $X = (x, y)$ y a la escala σ.

$$H(X, \sigma) = \begin{bmatrix} L_{xx}(X, \sigma) & L_{xy}(X, \sigma) \\ L_{xy}(X, \sigma) & L_{yy}(X, \sigma) \end{bmatrix}$$

$L_{xx}(X, \sigma)$ hace referencia a la convolución de la derivada de la gaussiana de segundo orden $\frac{\partial^2 g(\sigma)}{\partial x^2}$, con la imagen en el punto $X = (x, y)$, y de igual forma L_{yy} y L_{xy}. Estas derivadas son conocidas como *Laplacian of Gaussian (LoG)*. SURF, igual que *SIFT*, hace uso de estas derivadas para el cálculo de los puntos interesantes, y es por esto que se considera que SURF está parcialmente inspirado en el algoritmo SIFT. *David Lowe* [7] propuso aproximar el cálculo de las *LoG*, mediante una diferencia de gaussianas, *DoG (Difference of Gaussian)*, cuyo coste de cómputo es muchísimo menor. De la misma forma, *Herbert Bay* [21] propuso realizar este cálculo mediante el uso de filtros aproximados que representaran los *kernels* gaussianos utilizados. Esta última aproximación se comporta extremadamente bien junto con las imágenes integrales por lo que ofrece un rendimiento y velocidad muy elevados. A estos filtros se les llama D_{xx}, D_{yy} y D_{xy}.

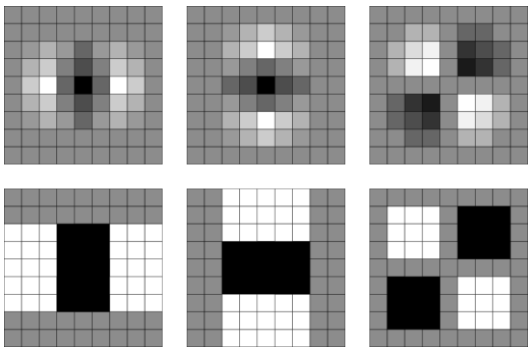

FIGURA 3.13: APROXIMACIÓN DEL OPERADOR LOG MEDIANTE BOX FILTERS. FILA SUPERIOR: DERIVADAS DE SEGUNDO ORDEN DE LA GAUSSIANA DISCRETIZADA (L_{XX}, L_{YY}, L_{XY}). FILA INFERIOR: APROXIMACION MEDIANTE FILTROS PONDERADOS (D_{XX}, D_{YY}, D_{XY})

En la Figura 3.13 se muestran los filtros utilizados por SURF para el cálculo de las derivadas parciales. Estos filtros se ponderan de diferente forma dependiendo de su direccionalidad. Para el filtro D_{xy} las zonas negras se ponderan con un valor 1, mientras que las zonas blancas se ponderan con un valor -1. Por otro lado, los filtros D_{xx} y los D_{yy} se ponderan asignando a las zonas blancas un valor -1 y a las zonas negras el valor 2. Las zonas de color gris sonde valor 0. Mediante el uso de esta aproximación, *Bay* propuso el cálculo del determinante de la matriz Hessiana de la siguiente forma:

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2$$

Esta aproximación del determinante es muy precisa y muy rápida de calcular, por lo que la pequeña pérdida de exactitud se ve compensada con la gran velocidad de cómputo.

En conclusión, el determinante de la matriz $H(X, \sigma)$ hace referencia a la respuesta del algoritmo en un punto $X = (x, y)$ a una determinada escala σ. El procedimiento seguido por SURF para detectar los puntos interesantes consiste pues en calcular dicho determinante, aproximado mediante los filtros gaussianos, para cada píxel y para varias escalas en cada píxel. Una vez calculados estos valores, se toman los máximos obtenidos para cada píxel en el espacio de escalas, y se consideran sólo puntos interesantes aquellos que superen un determinado umbral introducido como parámetro del método.

Espacio de escalas

Con el fin de que SURF constituya un algoritmo de extracción de características invariante a escala, es necesario definir el concepto *espacio de escalas*. El *espacio de escalas* es una función continua utilizada para analizar la imagen a diferentes escalas y obtener los extremos locales mediante el *DoH* en cada punto. En visión por computador, el *espacio de escalas* se suele representar como una pirámide en la que la imagen original se va sub-sampleando mediante convoluciones iterativas con un *kernel* gaussiano apropiado. Sin embargo SURF realiza una aproximación diferente del espacio de escalas. En vez de reducir o ampliar el tamaño de las imágenes, el algoritmo reduce y amplia el tamaño del filtro que se aplica a la imagen para calcular las respuestas del *DoH* en las diferentes escalas. Esta aproximación tiene la ventaja de evitar el efecto *aliasing* ya que no se realiza sub-sampling de la imagen.

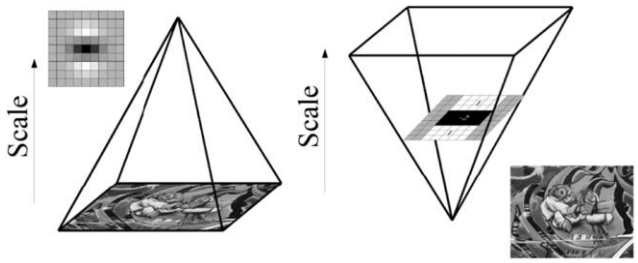

FIGURA 3.14: ESPACIO DE ESCALAS. IZQUIERDA APROXIMACION TRADICIONAL CON ESCALADO DE LA IMAGEN. DERECHA APROXIMACION SURF CON ESCALADO DEL FILTRO

El espacio de escalas se divide en *octavas*. Una *octava* constituye una serie de filtros o respuestas del algoritmo que engloban un factor de escala de dos. En la Figura 3.15 se presenta la relación entre las escalas y las octavas. La primera octava engloba los filtros $9 \times 9, 15 \times 15, 21 \times 21$ y 27×27, la segunda $15 \times 15, 27 \times 27, 39 \times 39$ y 51×51, etc.

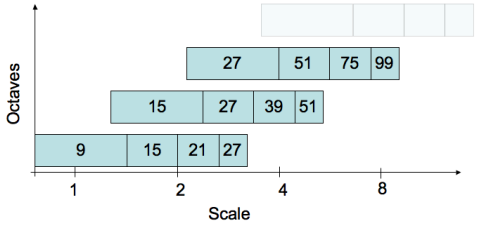

FIGURA 3.15: ESPACIO DE ESCALAS Y OCTAVAS

El mínimo factor de escala que considera SURF es el obtenido por el filtro de 9×9 presentado anteriormente, que se corresponde con una gaussiana real de $\sigma = 1.2$. Los siguientes niveles de escalado se obtienen aumentando los filtros y manteniendo este ratio inicial. En definitiva, las nuevas escalas se calculan de la siguiente forma:

$$\sigma_{appox} = Current\ FilterSize \times \frac{Base\ Filter\ Scale}{Base\ Filter\ Size} = Current\ Filter\ Size \times \frac{1.2}{9}$$

Localización de los puntos de interés

Resumiendo, la detección de puntos interesantes invariantes a escala y rotación en una imagen, se divide en tres etapas. En primer lugar se calcula la matriz Hessiana a varias escalas para cada píxel de la imagen y se filtran las respuestas ofrecidas por el *DoH* que no superen un *threshold* determinado. El

algoritmo elimina todos aquellos valores que se encuentran por debajo de este *threshold* por lo que a *thresholds* mayores, menor cantidad de puntos interesantes se obtiene de la imagen.

En segundo lugar, se aplica un proceso de búsqueda de máximo local (*Non-maximum suppression)* sobre cada una de las respuestas filtradas. Cada píxel en el espacio de escalas se compara con 26 vecinos formados por los 8 colindantes de la escala nativa del punto y los 18 (9 y 9) referentes a las escalas superior e inferior a la que se detectó el punto. En la Figura 3.16 se observa el proceso de *Non-Maximum Suppression*. Los círculos verdes representan los 26 vecinos contra los que se compara el punto analizado. El píxel marcado con una "X" se considerará máximo local si su valor es mayor que los 26 vecinos descritos.

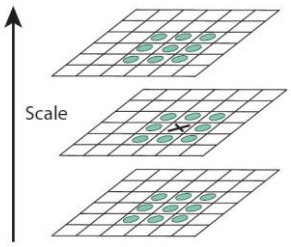

FIGURA 3.16: LOCALIZACIÓN DEL PUNTO DE INTERÉS. NON-MAXIMUM SUPPRESSION

El último paso consiste en la interpolación de la información de espacial de estos 26 vecinos para ajustar la localización y la escala óptima del punto. Este proceso se realiza expresando el determinante de la matriz Hessiana, $H(x, y, \sigma)$ como una serie de *Taylor* de orden cuadrático.

En conclusión, tras este proceso, SURF genera un conjunto de puntos interesantes de la imagen con unas propiedades de repetibilidad y precisión elevadas. Dichos puntos son robustos, invariantes a rotación y escala y de un alto grado de representabilidad de la imagen.

DESCRIPCIÓN DE PUNTOS INTERESANTES

Los descriptores proporcionados por el algoritmo SURF, describen cómo se distribuyen las intensidades de los píxeles cercanos al punto interesante, contenidos en un espacio determinado por la escala de dicho punto. Estas respuestas se obtienen mediante unos filtros concretos llamados *wavelets de Haar*. Las *wavelets de Haar* son filtros muy sencillos que se utilizan para buscar cambios de gradientes en las direcciones x e y. Estos filtros se ponderan asignando un valor 1 a la zona negra y -1 a la zona blanca.

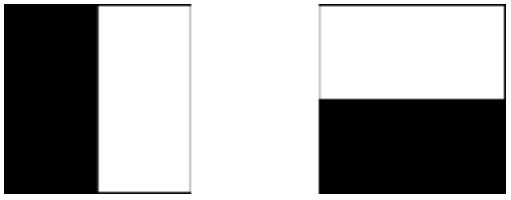

FIGURA 3.17: WAVELET DE HAAR. DERECHA DIRECCION X. IZQUIERDA DIRECCION Y

Orientación

Con el objetivo de que el algoritmo sea invariante a rotación, a cada punto interesante se le asigna una orientación concreta. La descripción posterior del punto se realiza en base a esta orientación por lo que es necesario que este valor sea reproducible bajo multitud de condiciones. Para determinar

esta orientación, se calculan las *wavelets de Haar* sobre un área circular de 6σ alrededor del punto de interés, donde σ hace referencia a la escala sobre la que se detectó dicho punto. Estos valores se ponderan posteriormente con una gaussiana centrada en el punto, dependiente de la escala y con una desviación típica de 2.5σ. Una vez ponderados estos valores, se proyectan en un espacio bidimensional en el que las respuestas en la dirección x del filtro de *Haar* se representan en las abscisas, y las repuestas en la dirección y se representan en las ordenadas. La orientación dominante se calcula rotando una ventana de $\frac{\pi}{3}$ sobre el origen y sumando las repuestas en x e y de todos los valores que se encuentran en dicha ventana, formando así un vector orientación. Por último, se elige como orientación final, aquel vector cuyo módulo es el mayor. En la Figura 3.18 se ilustra el procedimiento.

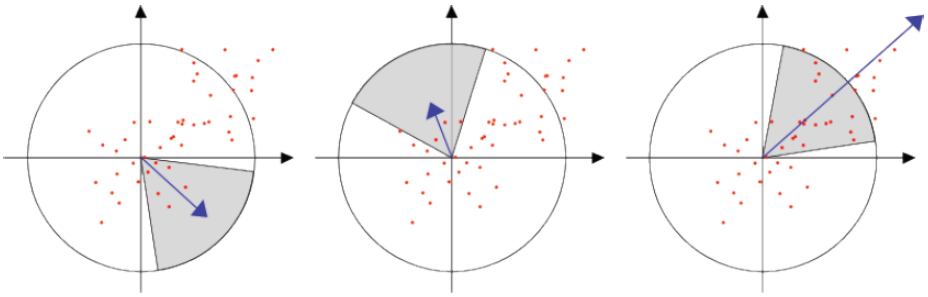

FIGURA 3.18: CÁLCULO DE VECTORES ORIENTACION

El objetivo de este proceso es construir descriptores invariantes a rotación. En algunas tareas, este requisito no es estrictamente necesario por lo que es posible omitir este paso e incrementar así las prestaciones del algoritmo. En definitiva, mediante este proceso se asigna a cada punto de interés una orientación que describe la dirección en la que los cambios de los gradientes cercanos al punto son más pronunciados.

Descriptor de puntos

El primer paso en la creación de los descriptores de los puntos consiste en la construcción de una ventana rectangular de tamaño 20σ, centrada en el punto interesante, con σ la escala del punto. Esta ventana contendrá los píxeles que formarán las entradas del descriptor. La ventana se subdivide en regiones de 4×4, sobre las que se calcula de nuevo las *wavelets de Haar* para 25 puntos de muestreo distribuidos regularmente sobre cada una de las sub-regiones. Si nos referimos a las respuestas obtenidas en x e y como dx y dy, entonces para los 25 puntos anteriores tenemos:

$$v_{region} = \left[\sum dx\,, \sum dy, \sum |dx|, \sum |dy|\right]$$

En consecuencia, cada región contribuye con cuatro valores al descriptor final. Dado que la ventana se divide en regiones de 4×4, tenemos que el vector final está formado por un total de $4 \times 4 \times 4 = 64$ valores.

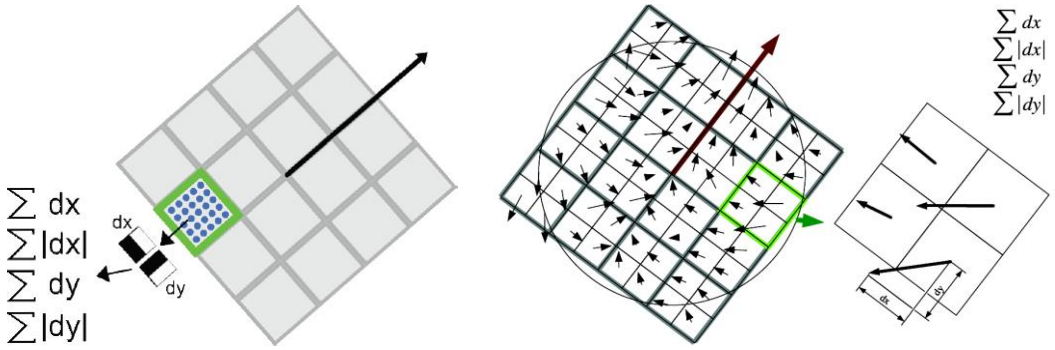

FIGURA 3.19: COMPONENTES DEL DESCRIPTOR SURF

En la Figura 3.19 se muestra el proceso anteriormente comentado. La ventana se subdivide en regiones de 4×4. El cuadrado de color verde se corresponde con una de las 16 sub-regiones y los puntos azules representan los 25 puntos de muestreo sobre los que se computan las *wavelets de Haar*. Como se observa, estas *wavelets* se calculan relativas a la orientación del punto, es decir, los filtros en las direcciones x e y están orientados en la dirección dominante. Mediante estos valores se construyen los descriptores 64-dimensionales que definen la información relevante de cada punto y su entorno.

A continuación se presentan dos ejemplos de extracción de puntos del algoritmo SURF sobre una imagen de un logotipo y una página publicitaria.

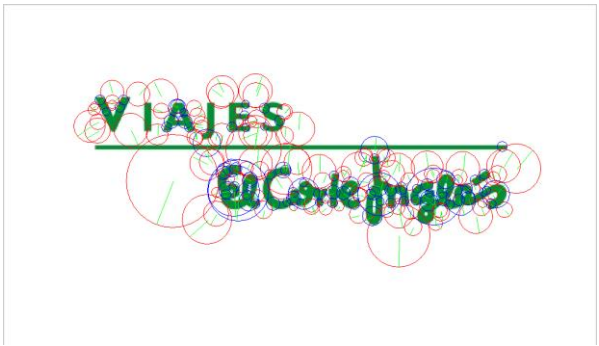

FIGURA 3.20: EJEMPLO EXTRACCIÓN DE PUNTOS SURF EN LOGOTIPO

FIGURA 3.21: EJEMPLO EXTRACCIÓN DE PUNTOS SURF EN PÁGINA PUBLICITARIA Y DETALLE.

Cada circunferencia se corresponde con un punto detectado. Se puede observar que existen circunferencias de diferentes tamaños. Estos tamaños son relativos a la escala a la que se ha detectado el punto y el diámetro de la circunferencia engloba la información espacial del entorno que se ha descrito para ese punto. También se observan líneas que parten del centro del círculo y finalizan en un punto límite de la circunferencia. Estas líneas indican la orientación del punto que anteriormente se ha explicado. Por último se pueden observar círculos cuyo borde esta dibujado en rojo y otros cuyo borde esta dibujado en azul. Esta diferencia se debe al signo del operador *laplaciano* calculado durante el proceso de detección de puntos. Básicamente se corresponde con la naturaleza del brillo del punto.

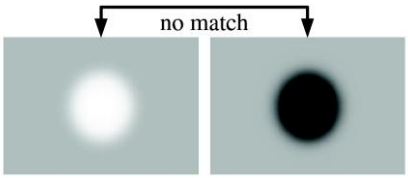

FIGURA 3.22: RELACIÓN DE BRILLO ENTRE FONDO Y OBJETO. OPERADOR LAPLACIANO.

El signo del operador *laplaciano* indica si el tipo de punto se corresponde con *mancha* clara sobre fondo oscuro, o al revés. Por la definición de los descriptores, el caso que se presenta en la Figura 3.22 no realizará *matching* nunca. Una forma rápida de acelerar el proceso de *matching* consiste en evaluar en primera instancia el signo operador *laplaciano* entre ambos puntos, y comprar si no son del mismo signo. En ese caso no es necesario realizar la comparación de las 64 componentes del descriptor.

Por último se puede observar la *repetibilidad* del algoritmo. A simple vista se puede observar que se han detectado puntos muy similares tanto en el logotipo como en la zona de la página publicitaria en la que se encuentra dicho logotipo. La *repetibilidad* es uno de los factores más importantes de un detector de puntos, puesto que dicho valor es un estimador de la robustez y precisión del algoritmo de extracción de características.

PARÁMETROS

El algoritmo SURF consta de los siguientes parámetros que se estudian a continuación:

- *InitSample*: *Downsampling* inicial realizado a la imagen.
- *Octaves*: Número de octavas a analizar por el algoritmo.
- *Threshold*: Umbral de detección de puntos interesantes.
- *Rotation*: Cálculo de descriptores con rotación.
- *Extended*: Cálculo de descriptores extendidos.

Init Sample

El parámetro *InitSample* constituye un valor entero utilizado para realizar un subsampling inicial a la imagen que se va procesar en el algoritmo. Así pues, al comienzo del método se realiza un escalado de la imagen y en posteriores etapas se construyen los mapas de respuestas del detector de puntos en relación al valor indicado por este parámetro.

El objetivo del *InitSample* está principalmente relacionado con la eficiencia computacional del algoritmo. Puesto que se realiza un subsampling inicial a la imagen mediante este valor, es obvio que si el *InitSample* es muy elevado se reducirá mucho el tamaño de la imagen dificultando gravemente la extracción de puntos. Este valor suele ser un valor fijo típicamente 2. Por lo tanto, a valores más elevados de *InitSample*, mayor velocidad del algoritmo en la extracción de puntos, pero peores resultados se obtienen. Por otro lado, ya que se escala inicialmente la imagen, es necesario utilizar el *InitSample* para la construcción de los mapas de respuestas del detector de puntos.

Por último, puesto que el *InitSample* responde a un escalado de la imagen, el rango de valores posibles que puede tomar este parámetro es el conjunto de los números racionales positivos $\mathbb{Q}$. Es obvio que, aunque sea posible este conjunto de valores, no tiene sentido evaluar muchos de ellos. Para el estudio de optimización que se presentará a continuación, se ha tomado un rango de $[2, 9]$.

Octaves

El parámetro *Octaves* es el principal responsable de la invarianza a escala del algoritmo. Como se ha explicado anteriormente, SURF es un algoritmo invariante a escala por lo que no es necesario para correlacionar dos imágenes, que estas guarden la misma relación de escala. El parámetro *Octaves* indica cuántas octavas debe computar SURF y por lo tanto cuanta información espacial se recoge sobre cada punto de la imagen. Recordar que cada octava está formada por un conjunto de filtros de diferentes tamaños, desde $9 \times 9, 15 \times 15, 21 \times 21$, etc., por lo que a mayor número de octavas, mayor cantidad de filtros y en definitiva mayor información espacial se recoge de cada punto.

SURF computa como máximo 5 octavas con los siguientes 4 filtros por cada octava:

TABLA 3: OCTAVAS Y FILTROS POR OCTAVA

	Filter 1	Filter 2	Filter 3	Filter 4
Octave 1	9x9	15x15	21x21	27x27
Octave 2	15x15	27x27	39x39	51x51
Octave 3	27x27	51x51	75x75	99x99
Octave 4	51x51	99x99	147x147	195x195
Octave 5	99x99	195x195	291x291	387x387

Obviamente, cuantas más octavas se calculen mayor lentitud del algoritmo, sin embargo generalmente mejores resultados se obtienen.

Threshold

El parámetro *Threshold* constituye el umbral sobre el cuál se decide cuándo una respuesta del detector de puntos se considera punto interesante. Como se ha explicado anteriormente, es necesario que la respuesta de la matriz *Hessiana* supere el umbral o *Threshold* para que el punto se considere de interés. De esta forma, cuanto menor es el *Threshold* más fácil es superar dicho valor y por lo tanto mayor cantidad de puntos interesantes se detectan de la imagen. Es necesario encontrar un valor óptimo de este parámetro para no generar puntos en exceso o en defecto. Ambos extremos son perjudiciales puesto que, generar puntos en exceso incrementa el coste computacional del sistema, mientras que no generar los puntos suficientes puede inducir a detecciones defectuosas o en el peor de los casos no llegar a detectar logotipos.

El rango de valores que puede tomar el *Threshold* se corresponde también con el conjunto de los números racionales positivos $\mathbb{Q}$, sin embargo el rango típico de valores que toma dicho parámetro se encuentra en torno a $[0.0001, 0.01]$.

Rotation

El parámetro *Rotation* indica mediante un *booleano* si se debe tener en cuenta la rotación o no, es decir, si los descriptores de los puntos deben calcularse atendiendo a esta posible transformación o la información espacial que se obtendrá alrededor del punto no será relativa a una posible rotación.

Existen multitud de tareas en las que se sabe a priori que los elementos que se desea detectar no aparecerán rotados. Si se posee este conocimiento, es posible indicar al algoritmo que no realice el proceso de cálculo de vectores orientación, rotación, etc., incrementando así la velocidad de cálculo de puntos interesantes. Para nuestra tarea, hemos asumido que los logotipos no aparecerán rotados, por lo que los experimentos se han realizado estableciendo dicho parámetro a *false*.

Extended

El parámetro *Extended* indica al algoritmo de extracción de características SURF, mediante un *booleano*, que compute descriptores de 128 características en vez de 64. Otros algoritmos como SIFT, calculan descriptores de 128 características recogiendo mayor información espacial alrededor del punto. Aunque los descriptores SURF de 64 características son, en la gran mayoría de los casos, suficientes para describir eficazmente la información espacial del punto, este parámetro permite al usuario obtener descriptores de mayor precisión asumiendo que la comparativa entre los mismos será más costosa. El sistema de reconocimiento de logotipos y marcas diseñado hace uso de descriptores SURF de 64 características locales.

En conclusión, el algoritmo SURF realiza un análisis minucioso de una imagen y construye un conjunto de descriptores relativos a los puntos interesantes detectados en la misma, de forma que estos descriptores sean invariantes a rotación, escala, cierta traslación y brillo. SURF construye descriptores vectoriales de 64 dimensiones que ofrecen una representación apropiada para una máquina y sobre los que se pueden aplicar fácilmente clasificadores inteligentes, como los presentados en el capítulo 3.2.3. El algoritmo consta de varios parámetros configurables sobre los que se puede actuar para obtener una respuesta orientada a eficiencia y velocidad o precisión y calidad de resultados

3.2.2.1 ESTIMACIÓN DE LOS PARÁMETROS ÓPTIMOS

En el anterior apartado, se ha presentado el funcionamiento del algoritmo SURF y sus parámetros asociados. Estos últimos nos permiten definir el funcionamiento del sistema dependiendo de la respuesta que busquemos del mismo. Variando estos parámetros podemos priorizar la velocidad y el coste computacional, por ejemplo orientado a dispositivos de menor potencia como móviles o sistemas embebidos, o a precisión y calidad de los resultados, para sistemas de mayor potencia.

En este punto, es necesario diseñar una metodología que nos permita obtener dichos parámetros teniendo en cuenta todas las restricciones impuestas por las necesidades de la tarea.

FUNCIÓN OBJETIVO

Se ha enfocado el problema de la búsqueda de la configuración óptima a través de la maximización de una función objetivo, definida en base a los requisitos funcionales del sistema desarrollado en el proyecto REIMED. Así pues, el problema se formaliza de la siguiente forma:

$$confOpt = \underset{\substack{is \in IS \\ o \in O \\ thr \in Thr}}{\operatorname{argmax}} \Big(TargetFunction\big(Analysis(is, o, thr)\big)\Big)$$

Partiendo de esta generalización, en primer lugar es necesario definir una función objetivo que responda a los resultados que se desean conseguir del sistema. La función objetivo diseñada es la siguiente:

$$TargetFunction = Performance + Cost$$

$$Performance = Cases\ Perf + Points\ Perf$$

$$Cases\ Perf = 0.5 * \left(\left(\frac{1}{9} * Specificity\right) + \left(\frac{8}{9} * Sensitivity\right)\right)$$

$$Points\ Perf = 0.35 * \left(\frac{TPP}{TPP + FPP}\right)$$

$$Cost = 0.15 * \left(1 - \left(\frac{time - minTime}{maxTime - minTime}\right)\right)$$

donde:

$$Specificity \equiv \frac{TN}{TN + FP} \qquad Sensitivity \equiv \frac{TP}{TP + FN}$$
$$TPP \equiv TruePositivePoints\ (Puntos\ realmente\ bien\ matcheados)$$
$$FPP \equiv FalsePositivePoints(Puntos\ matcheados\ fuera\ del\ logotipo)$$
$$time \equiv Tiempo\ en\ computar\ la\ configuracion$$
$$maxTime \equiv Tiempo\ de\ la\ configuración\ más\ lenta$$
$$minTime \equiv Tiempo\ de\ la\ configuración\ más\ rápida$$

En primer lugar observamos la ecuación general de la función objetivo. Podemos ver que se divide en dos grandes bloques: el rendimiento (*Performance*) y el coste (*Cost*). El bloque *Performance* hace referencia a la calidad y precisión de los resultados, mientras que el bloque *Cost* evalúa el coste temporal del algoritmo. Ambos factores están ponderados con respecto a la importancia que hemos considerado atendiendo a los requisitos funcionales definidos en el proyecto. Hemos ponderado el *Performance* con un 85% de la importancia de la función objetivo, mientras que al coste le hemos dado

un 15% de relevancia. No están ponderados explícitamente en esta ecuación pero sí en las sucesivas que se despliegan del desarrollo de las mismas. De esta forma premiamos a las configuraciones que obtengan mejores resultados a pesar de que su coste temporal sea superior.

Centrándonos en el *Performance*, el impacto de dicho factor se divide en dos bloques: el rendimiento en casos (*Cases Perf*) y el rendimiento en puntos (*Points Perf*). El motivo de esta división deriva de la cantidad de puntos que constituyen un *matching*. Debido a las condiciones de resolución, calidad de la imagen, escalado, etc. se dan casos en los que un *matching* puede estar formado por muchos puntos o por muy pocos. Si únicamente nos centráramos en la cantidad de puntos bien *matcheados,* conforme evaluamos configuraciones sub-óptimas, podríamos observar un descenso de este número de puntos. Sin embargo no podríamos especificar si ese descenso se debe a que los *matchings* se realizan con menos puntos o a que estamos perdiendo casos completos de detección. Imaginemos un caso en el que tenemos una configuración A con un total de 60 puntos *matcheados* correctamente. Esta cantidad de puntos se reparte en dos casos, un *matching* de 55 puntos y otro de 5. Si ahora evaluamos otra configuración B y nos fijamos únicamente en la cantidad de puntos bien *matcheados*, por ejemplo 50, no podemos asegurar que se sigan detectando los dos mismos casos. Es posible que en la configuración B se haya pasado a detectar dos casos de por ejemplo 46 y 4 puntos *matcheados*, o estemos perdiendo completamente el caso anterior de 5 puntos y el *matching* actual sea de un solo caso que ha pasado de tener 60 puntos correlacionados a 50. Por este motivo, es necesario separar el *Performance* a nivel de casos detectados y a nivel de puntos bien correlacionados.

Por otro lado, el factor *Cases Perf* evalúa la *sensibilidad* y *especificidad* del sistema. Una de las restricciones definidas en las especificaciones funcionales del proyecto, establece que es preferible hasta 8 veces más *falsos positivos* que 1 *falso negativo*. La *sensibilidad* nos mide el ratio de *verdaderos positivos* alcanzado, o visto de otra forma, invirtiendo probabilísticamente este valor tenemos el ratio de *falsos negativos* que hemos producido. En definitiva, dada una *especificidad* alta, significa que estamos generando muy pocos *falsos negativos* por lo que dicho valor debe ser 8 veces más importante $\left(\frac{8}{9}\right)$ que la *especificidad* para cumplir con la restricción. La *especificidad* sin embargo mide lo contrario. Con este estimador medimos la relación de la cantidad de *falsos negativos* que hemos generado, con respecto al conjunto de resultados generales. Según la regla establecida, se permiten hasta 8 veces más *falsos positivos* que *falsos negativos* por lo que el factor de importancia de este estimador debe ser menor $\left(\frac{1}{9}\right)$. Para finalizar se ha ponderado este coeficiente con un 50% de relevancia sobre la función objetivo puesto que es prioritario detectar el mayor número de casos posibles.

Por otro lado tenemos el factor *Points Perf*. Dicho factor estima la cantidad de puntos bien correlacionados, es decir, la cantidad de puntos situados correctamente sobre el logotipo. El conjunto de puntos TPP $(True\ Positive\ Points)$ representa aquellos puntos que se sitúan sobre los logotipos, mientas que el conjunto FPP $(False\ Positive\ Points)$ representa aquellos puntos que por causas locales específicas de la página se sitúan fuera de los logotipos. La suma de ambos conjuntos constituyen el total de puntos que se han *matcheado* en el análisis. Podemos obtener un ratio entre el conjunto TPP y el total que nos indica la fiabilidad del sistema. Cuanto más alto sea este ratio, mejor será la configuración y más contribuirá al valor final de la función objetivo. Finalmente se ha ponderado este factor con un 35% de relevancia sobre la función objetivo ya que, como se ha comentado anteriormente, es deseable los máximos puntos bien correlacionados posibles, pero siempre atendiendo más al mayor número de detecciones totales.

Por último queda el coste del sistema. Debido a las especificaciones de *AuditMedia*, el coste computacional y temporal del sistema no constituye un factor decisivo en el diseño, por lo que se ha optado por ponderarlo con un valor bajo. Este factor estima el tanto por ciento de proximidad del

tiempo en la configuración actual, con respecto al tiempo de la configuración más costosa. El factor *Cost* realiza una normalización del tiempo entre 0 y 1 para estimar en que intervalo se situaría dicho tiempo con respecto al total del rango observado. Es fácil ver que para el menor de los tiempos el ratio es 0 y para el mayor de los tiempos el ratio es 1. Puesto que buscamos la maximización de la función objetivo, un valor bajo de este ratio indica un comportamiento deseable. Por este motivo es necesario invertir dicho ratio, restando $1 - ratio$, para convertir un valor bajo en un valor alto y así incrementar la función objetivo en estos casos.

Por motivos de coste computacional de los experimentos, los análisis sobre el *Phantom* se han realizado comparando dicha imagen contra exactamente los 41 logotipos existentes en el mismo. Esto conlleva que no exista tasa de *verdaderos negativos* (TN), es decir, si se emplea una base de datos de logotipos que contiene exactamente los mismos elementos existentes en una sola página (*Phantom*), es imposible que exista un *verdadero negativo.* Por este motivo no es posible calcular la *especificidad* en la función objetivo. Para suplir este inconveniente, se ha simplificado la parte correspondiente al cálculo de la *especificidad*, realizando esta pequeña modificación del coeficiente $Cases\ Perf$:

$$Cases\ Perf\ = \ 0.5 * \left(\left(\frac{1}{9} * \left(1 - \left(\frac{FP}{MaxFP} \right) \right) \right) + \left(\frac{8}{9} * Sensitivity \right) \right)$$

donde

$$MaxFP = Nº\ objects\ \times Nº\ imagenes$$

Así pues, se considera la función objetivo general para los casos de análisis de bases de datos reales, mientras que esta pequeña simplificación se aplica únicamente a los experimentos realizados sobre le *Phantom.*

METODOLOGÍA

Definida la función objetivo, queda realizar un análisis de una base de datos concreta con varias configuraciones del algoritmo SURF. Una vez realizado el análisis para estas configuraciones, será necesario evaluar los resultados obtenidos por cada una de ellas mediante la función objetivo. Finalmente el objetivo será encontrar aquella configuración que maximice la función objetivo.

El análisis que se ha planteado ha consistido en evaluar 1000 configuraciones posibles de SURF. Para ello se han evaluado los 41 logotipos de la base de datos *Multi-Medio* sobre el *Phantom*. El conjunto de configuraciones que se ha explorado en el análisis atiende a los siguientes rangos de valores:

- $8\ InitSamples\ desde\ [2, 9]$
- $5\ Octaves\ desde\ [1, 5]$
- $25\ Thresholds\ desde\ [0.0001 : 0.00041666666666 : 0.01]$

En total $8\ InitSamples\ \times 5\ Octaves\ \times 25\ Thresholds = 1000\ configuraciones$

Notar que para el conjunto de valores de *InitSample* se ha partido del valor 2 en vez de 1. El motivo de esta elección se corresponde con problemas de coste computacional. Se probó a analizar el *InitSample* 1 pero se tuvo que descartar debido a su elevadísimo coste computacional. Por otro lado, se estableció como cota superior de este parámetro el valor 9 puesto que a valores superiores se producían problemas de estabilidad y se abortaba la ejecución del algoritmo interrumpiendo el proceso

de análisis general. En cuanto a las *Octaves* se han analizado todas las posibles y en cuanto al *Threshold* se ha tomado dicho rango representativo sobre el cuál se sitúa con toda seguridad el valor óptimo.

Para finalizar, se ha lanzado dicho experimento en la plataforma MATLAB® y sobre la librería IRIS, implementada en C#. Puesto que ambas plataformas difieren tanto en costes computacionales como en detalles de implementación, ha sido necesario estimar estos parámetros para ambos sistemas.

3.2.3 CLASIFICACIÓN (MATCHING)

3.2.3.1 CLASIFICADOR DE MÍNIMA DISTANCIA.

El clasificador que se ha utilizado para el proceso de *matching* entre puntos de los logotipos y las páginas publicitarias es un clasificador de mínima distancia. Este clasificador no se corresponde con la definición formal dada en el capítulo de introducción, más bien se puede interpretar como un algoritmo de comparación de puntos por distancia euclídea gobernado por un determinado umbral. El motivo reside en que no existen varias clases sobre las que decidir a cual pertenece el objeto, sino que simplemente cada punto del logotipo debe asociarse con el punto más cercano. Este clasificador por lo tanto, no necesita de entrenamiento previo y no define fronteras de decisión sobre ningún plano, ni funciones discriminantes. Aun así, este clasificador podría formalizarse como un *k-vecinos* con $k = 1$ y con un número de clases equivalente al número de puntos extraídos de la página publicitaria.

El clasificador de mínima distancia recibe como parámetros de entrada, los descriptores de los puntos extraídos del logotipo y de la página publicitaria respectivamente. El clasificador realiza un proceso iterativo en el que toma secuencialmente descriptores de puntos del logotipo, y los compara contra todos los descriptores de los puntos de la página publicitaria. Para cada par de descriptores 64-dimensionales, el clasificador calcula la distancia euclídea entre ambos, y almacena los dos mejores candidatos de la página publicitaria.

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_{64} - y_{64})^2}$$

En otras palabras, el clasificador compara punto a punto los descriptores del logotipo con los de la página publicitaria y guarda para cada punto del logotipo, los dos puntos de la página publicitaria que menor distancia euclídea presentan. Una vez obtenidos estos dos puntos, se aplica una sencilla regla para decidir si el punto más cercano de los dos se considera interesante o no. La regla es la siguiente: Dado un punto del logotipo p_{logo}, y los dos puntos más cercanos de la página publicitaria p_1 y p_2, con $d(p_{logo}, p_1) < d(p_{logo}, p_2)$:

$$si\ \frac{d(p_{logo}, p_1)}{d(p_{logo}, p_2)} < 0.65\ , entonces\ Match(p_{logo}, p_1) = true$$
$$sino \qquad\qquad Match(p_{logo}, p_1) = false$$

Donde el umbral 0.65 ha sido establecido de forma empírica mediante una curva ROC. Antes de proseguir con la explicación del método haremos una diferenciación entre *match* y *matching*. Un *match* es una identificación positiva entre un punto del logotipo y un punto de la página publicitaria, mientras que un *matching* es la detección completa de un logotipo en una página publicitaria. Por lo tanto un *matching* está formado por un conjunto de 3 *matches* como mínimo.

Esta regla introduce un significado concreto. Básicamente lo que se pretende con dicha regla es discernir entre un *match* casual y un *match* correcto. Mediante esta regla se considera que el punto p_1 es un *match* correcto siempre y cuando exista una diferencia notable de distancia entre el mejor punto y el segundo mejor punto. Si la $d(p_{logo}, p_1)$ es bastante menor que la $d(p_{logo}, p_2)$ entonces se considera

que p_1 es un *match* correcto. La Figura 3.23 muestra esta relación entre las distancias entre los dos mejores puntos.

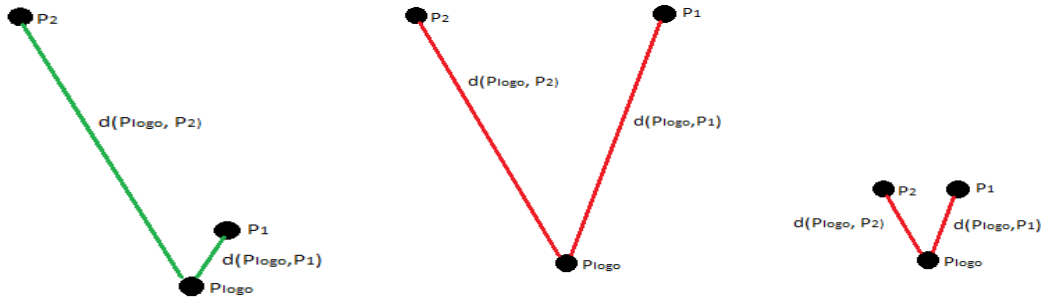

FIGURA 3.23: DISTANCIAS MATCHING MEJOR PUNTO VS SEGUNDO MEJOR PUNTO. IZUQUERDA MATCHING CORRECTO. DERECHA Y CENTRO MATCHINGS INCORRECTOS.

El único inconveniente de esta regla reside en el mostrado en la imagen de la derecha de la Figura 3.23. En este caso, ambos puntos se encuentran a una distancia muy pequeña con respecto al punto de interés, sin embargo ninguno de los dos se considerará *match* al no guardar la relación de distancias requerida. Se podría pensar que, aunque no se guarde dicha relación, si las distancias son muy pequeñas entonces se considere *match* al menor de los dos puntos, pero esto lleva a un error. En primer lugar hay que pensar que estas distancias no son distancias espaciales, es decir, las distancias representan la diferencia entre los 64 valores del descriptor de un punto contra los 64 valores del descriptor de otro punto. Por lo tanto, que ambas distancias sean similares no implica que ambos puntos se encuentren cercanos espacialmente en la imagen. Por otro lado, una distancia pequeña no implica que el *match* sea correcto. Es posible que por condiciones morfológicas de la página o del entorno, se encuentre un punto aislado que obtenga una mejor distancia que el punto correcto que se busca. Para finalizar, mediante esta regla se premia los *matches* más rigurosos, es decir, se premia las situaciones en las que se encuentra un mejor punto muy correlacionado y un segundo mejor punto bastante diferente del punto en cuestión.

En último lugar, consideramos un *matching* completo, es decir, una detección positiva de un logotipo en una página publicitaria, si se consigue correlacionar como mínimo 3 puntos. A priori se puede pensar que un *matching* de 3 puntos es muy poco relevante, sin embargo debido a defectos de compresión, resolución o escalado de la página, es frecuente encontrar logotipos en una calidad baja. Es difícil pues *matchear* más de 3 o 4 puntos sobre estos logotipos, por lo que si aumentamos esta cota perderíamos detecciones correctas de anuncios.

3.2.3.2 OPTIMIZACIÓN

ESTRATEGIA VORAZ

La optimización que se presenta deriva de la redundancia al comparar puntos previamente *matcheados* en iteraciones anteriores. El problema es el siguiente: En el proceso de clasificación de los puntos, el clasificador de mínima distancia toma cada uno de los puntos del logotipo y los compara contra todos los puntos extraídos de la página publicitaria. Dado un *matching* de un punto en una iteración determinada, en la siguiente iteración se volverá a evaluar este mismo punto contra otro distinto del logotipo en busca de otro posible *matching*. Esta repetición da muchos problemas puesto que se *matchean* diferentes puntos de un logotipo contra el mismo punto en la página publicitaria.

FIGURA 3.24: EJEMPLO REAL ESTRATEGIA VORAZ

En la Figura 3.24 se muestra un caso real del problema inducido por la redundancia en la comparación de puntos. Como se puede observar, se ha conseguido hacer un *matching* de un total de 13 puntos del logotipo, pero todos ellos se han correlacionado con el mismo punto en la página publicitaria.

La estrategia que se ha adoptado para solucionar este problema es una estrategia de tipo voraz. Consiste en, una vez conseguido correlacionar un punto del logotipo con su mejor punto de la página publicitaria, este último se extrae del conjunto de puntos de la página y ya no se vuelve a evaluar jamás. Esta estrategia, en contra de lo que puede parecer a priori, ayuda a la detección de *matchings* correctos y genera detecciones de mejor calidad. Esto es debido a que obviamente los puntos interesantes extraídos en las zonas de la página publicitaria en las que se encuentra el logotipo, son puntos muy similares a los que se han extraído del propio logotipo. Por lo tanto, será obvio que cada punto del logotipo se asociará con su homólogo correspondiente en la página publicitaria y no se repetirá ninguno.

Por el contrario, en el caso en el que no exista el logotipo en la página si aplicamos la estrategia voraz ocurrirá únicamente realizará *matching* un punto y ya ninguno de los puntos siguientes se evaluarán contra el mismo. De esta forma, sobre el ejemplo anterior, reducimos el caso de 13 puntos incorrectos a 1 punto incorrecto. Teniendo en cuenta la regla definida en el apartado anterior, en la que consideramos una detección de logotipo aquel *matching* de al menos 3 puntos o más, habremos conseguido eliminar un *falso positivo* de forma automática.

Por otro lado, esta optimización incrementa la velocidad de comparación de puntos. Conforme aumentan los puntos correlacionados, se van extrayendo estos del conjunto de puntos total de la página publicitaria, por lo que el tiempo de cálculo de las comparaciones disminuye. El número de puntos extraídos tanto de logotipos como de páginas publicitarias, está en función de los parámetros del algoritmo SURF. Para la configuración óptima de la plataforma MATLAB® ($initSample = 2, octaves = 4, threshold = 0.0005$), la media de puntos extraídos de un logotipo calculada a partir del conjunto de logotipos de la base de datos *Multi-Medio* de 41 logotipos es $N \approx 180$ puntos. La media de puntos extraídos de una página publicitaria, calculada sobre las 40 páginas de la base de datos *Multi-Medio* es $M \approx 12000$ puntos. El número de comparaciones necesarias sin la estrategia voraz es $C \approx N \times M \approx 180 \times 12000 \approx 2160000$ comparaciones. Suponiendo que de los 180 puntos del logotipo se consigue realizar una media de 15 *matches*, el número de comparaciones de punto a punto se reduce a:

$$C \approx 165 \times 12000 + \sum_{i=1}^{15} 12000 - i \approx 2159880$$

El decremento en el número de comparaciones no es ni mucho menos significativo, incluso insignificante con respecto al total de comparaciones naturales, sin embargo es un decremento y como tal es una optimización. Además esta optimización se ve maximizada cuantos más puntos se consiguen correlacionar, por lo que conseguir un alto índice de correlación entre un logotipo y una página publicitaria, además de ofrecer una mayor relevancia y seguridad estadística, implica disminuir gradualmente el tiempo de comparación entre ambos.

3.2.4 CLASIFICACIÓN (POST-PROCESO)

3.2.4.1 VARIABLES DISCRIMINANTES

Para la discriminación entre *falsos positivos y verdaderos positivos* ha sido necesario estudiar la naturaleza de ambos conjuntos. Generalmente un *verdadero positivo* suele estar formado por un número elevado de puntos mientras que un *falso positivo* consta de muy pocos *matchings*, sin embargo existen casos en los que esto no siempre se cumple. Por cuestiones de calidad de las imágenes, compresión o resolución, existen casos de detecciones correctas con un número mínimo de puntos *matcheados*. Estos casos se llaman *casos límite* y están formados por entre 3, 4 o 5 puntos. Por otro lado, un *falso positivo* suele contar también con 3 o 4 puntos, por lo que en definitiva, el objetivo consiste en diferenciar los *casos límite* correctos de los *falsos positivos*.

Para abordar la problemática de los *casos límite,* es necesario combinar la información ofrecida por el número de puntos correlacionados con otras variables que estimen de alguna forma la pertenencia del *matching* a una de las dos clases. Para ello, se ha diseñado un método que permita analizar las características morfológicas del *matching* y su similitud tanto en el logotipo como en la página publicitaria. Sobre este método se han ideado diferentes variables que permitan representar estas diferencias y que combinadas con el número de puntos correlacionados puedan crear un modelo discriminante adecuado.

NÚMERO DE PUNTOS CORRELACIONADOS

Como se ha dicho anteriormente, generalmente un *matching* correcto se suele corresponder con un elevado número de puntos correlacionados. Es lógico que si realmente existe el logotipo en la página publicitaria, el grado de correlación sea elevado. *Matchings* por lo tanto de más de 10 puntos, se consideran con casi total seguridad *verdaderos positivos*. Sin embargo, ya se ha comentado también que las condiciones de calidad, resolución o ruido pueden complicar este proceso y disminuir el número de puntos correlacionados. Por este motivo, no es suficiente con evaluar únicamente esta variable y por lo tanto hace falta un análisis más profundo de las características del *matching*.

DISTRIBUCIÓN DEL RATIO DE DISTANCIAS

El método consiste en lo siguiente: Dado un *matching* N puntos, el sistema conoce cuáles son esos N puntos tanto en el logotipo como en la página publicitaria. En primer lugar se calculan todas las distancias entre los N puntos del logotipo y todas las distancias entre los N puntos de la página publicitaria. En este punto se han probado diferentes métricas de distancias desde la euclídea, *mahalanobis*, *city block distance* ($\mathcal{L}_1$), *chebychev*, *haming*, etc. Finalmente se ha elegido la distancia *city block* ($\mathcal{L}_1$) por ofrecer unas condiciones de separabilidad entre las clases mayor. Una vez calculadas todas las distancias $\mathcal{L}_1$ entre los pares de puntos, a continuación se toman iterativamente los pares de distancias correspondientes al mismo par de puntos en el logotipo y en la página, y se dividen entre sí. Por último, se comprueba que todos estos ratios de distancias guardan la misma relación, que en definitiva se corresponde con la escala a la que se ha encontrado el logotipo en la página publicitaria. Se ilustra un ejemplo del método:

FIGURA 3.25: RELACIÓN DISTANCIAS VERDADERO POSITIVO

En la Figura 3.25 se muestra un *matching* de 5 puntos en el que a la izquierda se encuentra el logotipo con los puntos del *matching* y su distribución de distancias, y a la derecha se muestra la porción de la página en la que se encuentra el logotipo con los puntos del *matching* y su distribución de distancias también. Notar que para el ejemplo se ha pintado la distancia euclídea sin embargo en la práctica se utiliza la distancia $\mathcal{L}_1$ como se ha comentado anteriormente. Como se puede observar, a pesar de que existe un punto desplazado, la distribución de distancias es prácticamente idéntica. Si se realizan las divisiones de los pares de distancias correspondientes a los mismos puntos, se observará que todas guardan una relación común: la escala. Obviamente esta relación no es completamente exacta. En el caso ideal, la distribución de los ratios de las distancias se modelaría como una *delta de Dirac* centrada en el valor de la escala, sin embargo debido a las pequeñas variaciones y desplazamientos de los puntos, la distribución se corresponde con una *Gaussiana* o una *Cauchy* centrada en la escala y con una pequeña desviación típica.

En la Figura 3.26 se muestra el caso contrario. En esta figura se presenta la distribución de distancias de un *falso positivo*. De la misma forma que en el caso anterior, a la izquierda se muestra el logotipo y la distribución de distancias de los puntos del *matching*, mientras que la derecha se muestra la página publicitaria y su distribución de los puntos. Se puede ver perfectamente que si dividiéramos los pares de distancias correspondientes a los mismos puntos, estos no guardarían el mismo ratio sino que cada uno presentaría un valor diferente.

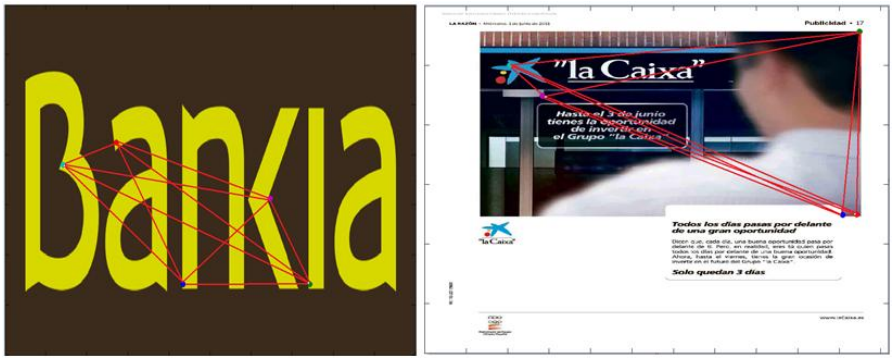

FIGURA 3.26: RELACIÓN DISTANCIAS FALSO POSITIVO

Así pues, el método consiste en obtener la distribución de ratios entre los pares de distancias de los puntos del *matching*, y determinar si siguen una distribución *Gaussiana*, o en el mejor de los casos si siguen una *delta de Dirac* y se agrupan con muy poca dispersión en torno a un mismo valor.

Para ello se han estudiado diferentes variables que puedan indicar el grado de dispersión de un conjunto de datos. Las variables son:

- El *rango inter-cuartil (IQR: Interquartile Range)*
- La diferencia entre la media y la mediana de la distribución
- El estimador *Median Absolute Deviation (MAD)*

Interquartile Range

El *rango inter-cuartil (IQR: Interquartile Range)* es un estimador robusto que mide la dispersión estadística que presenta un conjunto de datos que a priori siguen una distribución *Gaussiana*. El IQR es similar a la desviación típica, con la diferencia que este solo considera los datos que se encuentran entre el primer cuartil (Q1) y el tercer cuartil (Q3), donde se observa el 50% de los datos. De esta forma, el IQR es más robusto a *outlayers* y por lo tanto es capaz de determinar con mayor precisión si los datos se agrupan en torno a un mismo valor.

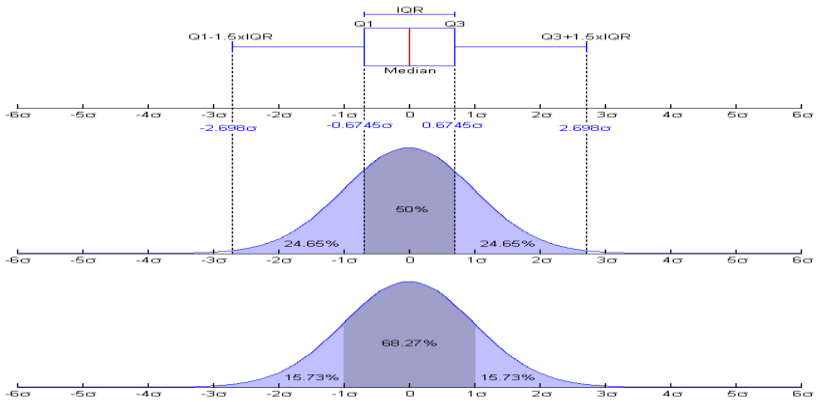

FIGURA 3.27: *INTERQUARTILE RANGE*

El IQR se calcula de la siguiente forma: Dada una distribución X de longitud $N = length(\mathrm{X})$ y con los valores ordenados de menor a mayor, el IQR es:

$$IQR = Q3 - Q1$$
$$Q3 = 75th\ percentil$$
$$Q1 = 25th\ percentil$$

Los Pth percentiles se calculan como:

$$n = round\left(\frac{P}{100} \times N + \frac{1}{2}\right)$$
$$Pth\ percentil = \mathrm{X}(n)$$

Dada la definición anterior de la distribución del ratio de distancias, es de esperar que si realmente existe el logotipo en la página, esta distribución esté centrada en un mismo valor y prácticamente sin dispersión alguna. En ese caso, para los *verdaderos positivos* el valor del IQR será muy bajo, cercano a 0, mientras que para los *falsos positivos* este valor crecerá muy rápido cuantos más puntos erróneos se hayan *matcheado*. Si combinamos esta variable con la cantidad de puntos que forman el *matching*, el objetivo a alcanzar consiste en obtener un alto valor del número de puntos correlacionados y un bajo valor de IQR. De esta forma podremos asegurar que el caso que se está analizando se corresponde con un *verdadero positivo*.

Diferencia entre la media y la mediana

La diferencia entre la media y la mediana es también un parámetro que nos puede indicar cuál es la dispersión que presentan los datos de una distribución. La media, como es bien sabido, es un estimador muy vulnerable a *outlayers*, mientras que la mediana se comporta mucho mejor frente a casos anómalos y por lo tanto ofrece una estimación mucho más robusta. Conociendo estas características, podemos afirmar que si la media y la mediana coinciden en el mismo valor, entonces es que la distribución no presenta mucha dispersión y los datos se centran en un valor común. No se expondrán ejemplos del cálculo de la media y la mediana puesto que se consideran conceptos básicos de sobra conocidos.

Combinando de igual forma, la cantidad de puntos del *matching* junto con esta diferencia de media y mediana, podemos construir otro modelo discriminante para diferenciar entre los *verdaderos positivos* y los *falsos positivos*.

Median Absolute Deviation (MAD)

El estimador MAD constituye otra medida robusta para el cálculo de la dispersión de los datos en una distribución univariada. El cálculo del MAD se realiza de la siguiente forma:

$$MAD = median(\ |\mathrm{X}(i) - median(X)|\ )$$

Este estimador calcula la mediana de la desviación absoluta de los valores de la distribución con respecto a la mediana de la distribución. Un ejemplo ilustra mejor el comportamiento de este estimador. Considérese el conjunto de datos $(1,1,2,2,4,6,9)$. La mediana de este conjunto es 2. La desviación absoluta con respecto a 2 es $(1,1,0,0,2,4,7)$. Si reordenamos el vector para el cálculo de la mediana, tenemos $(0,0,1,1,2,4,7)$ cuya mediana es 1. En conclusión, este estimador es más robusto que la desviación típica y nos ofrece otra variable adicional para la construcción del modelo discriminante.

3.2.4.2 Quadratic Discriminant Analysis. Mahalanobis distances

El clasificador *QDA* es el clasificador de Bayes *Gaussiano*. Este clasificador es una generalización del clasificador *LDA* (*Linear Discriminant Analysis)* puesto que si todas las clases comparten la misma matriz de covarianzas, entonces el término cuadrático se anula y el clasificador *QDA* se convierte en lineal. El clasificador *Gaussiano* es cuadrático con x y se define como:

$$c^*(x) = \arg\max_c g_c(x)$$

con

$$g_c(x) = x' W_c x + w_c^t x + w_{c0}$$

donde

$$W_c = -\frac{1}{2}\Sigma_c^{-1}$$
$$w_c = \Sigma_c^{-1}\mu_c$$
$$w_{c0} = \log p(c) - \frac{1}{2}\log|\Sigma_c| - \frac{1}{2}\mu_c^t\Sigma_c^{-1}\mu_c$$

El siguiente paso consiste en estimar los parámetros del clasificador, es decir, los valores de W_c, w_c y w_{c0}. Es necesario entrenar el clasificador con el modelo discriminante anteriormente definido y así obtener las fronteras de decisión de las clases. Para ello se ha hecho uso de la función *classify* de MATLAB® que implementa un análisis discriminante lineal o cuadrático. Los diferentes tipos de clasificadores posibles son:

- *linear*: Análisis discriminante lineal (*LDA*). Matriz de covarianzas común y completa para todas las clases.
- *diaglinear*: Análisis discriminante lineal. Matriz de covarianzas común y diagonal para todas las clases. Se asume que las variables son independientes y no existen correlaciones entre ellas (*naive Bayes classifiers*).
- *quadratic*: Análisis discriminante cuadrático (*QDA*). Matriz de covarianzas propia y completa para cada clase.
- *diagquadratic*: Análisis discriminante cuadrático. Matriz de covarianzas propia y diagonal para cada clase. Se asume que las variables son independientes y no existen correlaciones entre ellas (*naive Bayes classifiers*).
- *mahalanobis*: Análisis discriminante cuadrático (*QDA*). Matriz de covarianzas propias y completa para cada clase. Se ignora la probabilidad a priori de las clases. Hace uso de las distancias de *Mahalanobis*.

3.2.5 IMPLEMENTACIONES DE SURF

Se probarán diferentes implementaciones del algoritmo SURF con el objetivo de evaluar la que mejor rendimiento ofrezca.

Para el desarrollo del prototipo experimental, se hará uso de la implementación abierta de SURF para la plataforma MATLAB®, llamada OpenSURF escrita por *Dirk-Jan Kroon.* Esta implementación es una traducción optimizada del código escrito en C# por *Chris Evans*.

En cuanto al diseño de la librería final, se evaluará para C# la implementación oficial de OpenSURF llevada a cabo por *Chris Evans* que se puede encontrar en el siguiente *link*: http://www.chrisevansdev.com/computer-vision-opensurf.html. Esta es la implementación más popular de SURF y de mayor uso en la actualidad. Por último, también se evaluarán las implementaciones de la librería OpenCV http://opencv.willowgarage.com/wiki/ escritas en C++.

4 RESULTADOS

En este capítulo se presentan todos los resultados obtenidos en la experimentación de este proyecto. El capítulo se divide en dos secciones: Los resultados obtenidos por *Phase Correlation*, y los resultados obtenidos por SURF. Los primeros hacen referencia a los resultados obtenidos por el primer método de aproximación al problema que utilizamos. Estos resultados son simbólicos y se incluyen como documentación, sin embargo desechamos esta técnica muy rápidamente por lo que los resultados carecen de relevancia. Los resultados obtenidos por SURF son los resultados obtenidos por el sistema de reconocimiento de logotipos y marcas, ya que este es el algoritmo de extracción de características que utiliza el sistema. Por lo tanto, a través de estos resultados se evaluará las prestaciones y rendimiento del sistema.

4.1 PHASE CORRELATION

Se realizaron diversos experimentos para evaluar este método, de los cuales se presentan dos casos significativos. Se evaluó el logotipo de la Figura 4.1 y se comparó contra dos páginas de periódico, una en la que se encontraba dicho logotipo y otra en la que no existía.

FIGURA 4.1: EJEMPLO DE LOGOTIPO DE LA BASE DE DATOS MONO-MEDIO

FIGURA 4.2: EJEMPLO DE PÁGINAS CON LOGOTIPO EL LOGOTIPO DE EJEMPLO DE LA BASE DE DATOS MULTI-MEDIO. IZQUIERDA NEGATIVO, DERECHA POSITIVO

Para ambos casos, existen tres *variables* a despejar en el análisis: la escala del diccionario sobre la cual el método habrá encontrado la mayor correlación, la posición en la página y por último el propio índice de correlación.

En primer lugar analizaremos el caso positivo. En la Figura 4.3 se puede observar que en torno al índice de escala 180 se encuentra el pico de máximo valor, que se corresponde con un 70% de correlación. Además de este elevado grado de correlación, el pico notablemente diferenciado del resto de escalas lleva a pensar que no ha sido producto de la casualidad. Notar que el conjunto de escalas es decreciente, es decir, las escalas con índice menor corresponden con tamaños de la imagen mayores, esto es: $size(dicctionary\{1\}) > size(dicctionary\{2\} > \cdots > size(dicctionary\{N\})$.

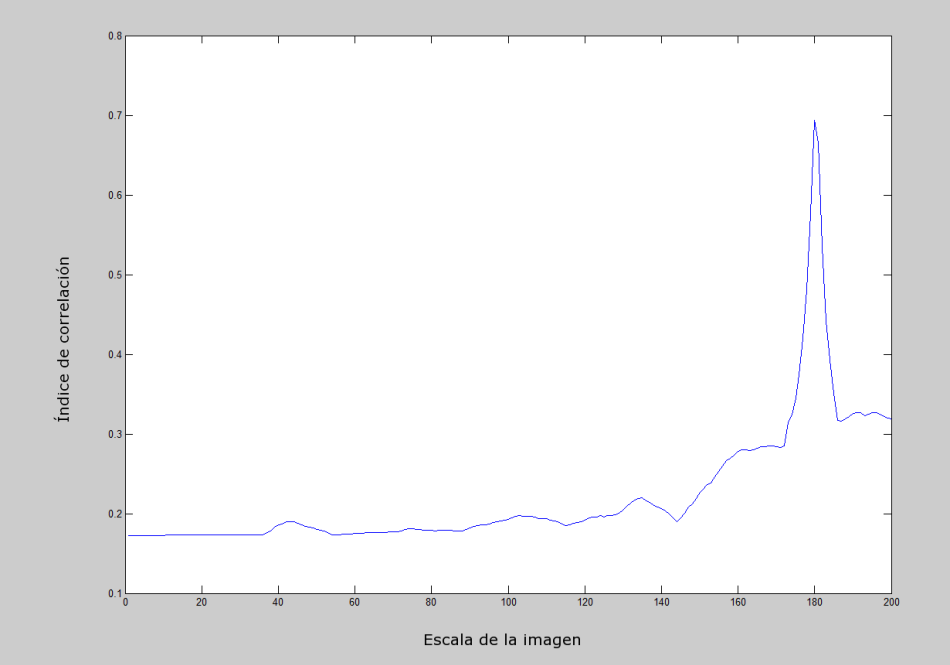

FIGURA 4.3: DICCIONARIO DE ESCALAS VS GRADO DE CORRELACIÓN PARA PÁGINA CON POSITIVO DEL LOGOTIPO

Para determinar el segmento de la señal que ha sido el causante del mayor grado de correlación, basta con realizar una gráfica del resultado local del análisis para cada píxel de la página de periódico. En la Figura 4.4 se observa como en la esquina inferior derecha, donde realmente se encuentra el logotipo, se localiza el pico más elevado de la función, que se corresponde con el 70% de correlación anteriormente mostrado.

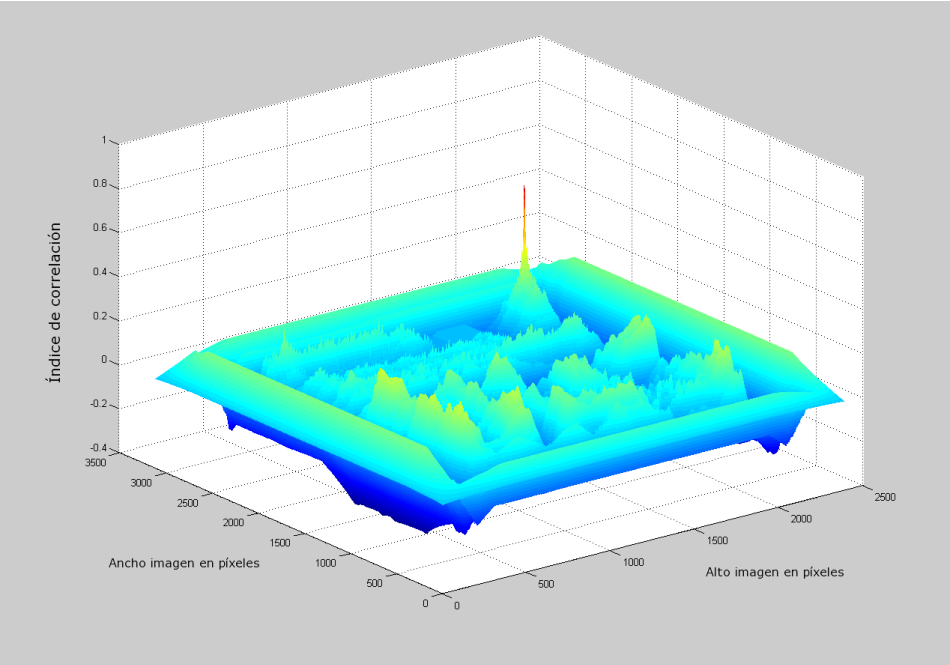

FIGURA 4.4: RESPUESTA POR PÍXEL DE PHASE CORRELATION PARA PÁGINA CON POSITIVO DEL LOGOTIPO

Si pasamos a analizar el caso negativo, los resultados del algoritmo son esclarecedores. De la misma forma que para el caso positivo, el análisis tratará de determinar el escalado que más se ajusta al posible logotipo, así como la posición del mismo y su grado de correlación. En la Figura 4.5 se observa como en este caso, para ninguna de las 200 escalas se ha sobrepasado un nivel de correlación del 32%. Este grado de correlación se encuentra en torno al índice de escala 165 y es producto de la casualidad o de los posibles *matchings* a nivel local. En contra del caso anterior, no se observa ningún pico diferenciado del resto sino que de forma más o menos uniforme existe un gran rango de escalas que obtienen un grado de correlación similar. Esto induce a pensar de forma correcta, que no existe el logotipo en la página.

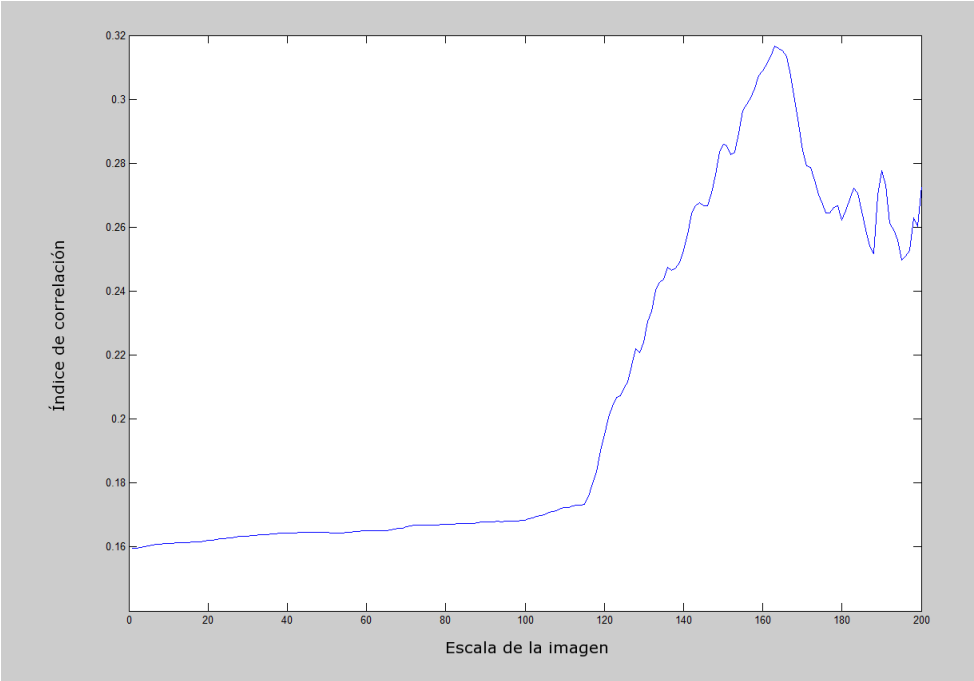

FIGURA 4.5: DICCIONARIO ESCALAS VS GRADO CORRELACIÓN PARA PÁGINA CON NEGATIVO DEL LOGOTIPO

Con respecto a la posición sobre la que se ha detectado el pico de mayor correlación, como se puede preveer, no existirá una región suficientemente diferenciada del resto. En la Figura 4.6 se observa la respuesta por píxel del algoritmo *Phase Correlation* para la página de periódico. Aunque se observa una pequeña región algo más diferenciada del resto, no es más de en torno a un 10% con respecto al resto de zonas de la página. En definitiva se puede determinar de forma correcta que no existe logotipo en esta imagen.

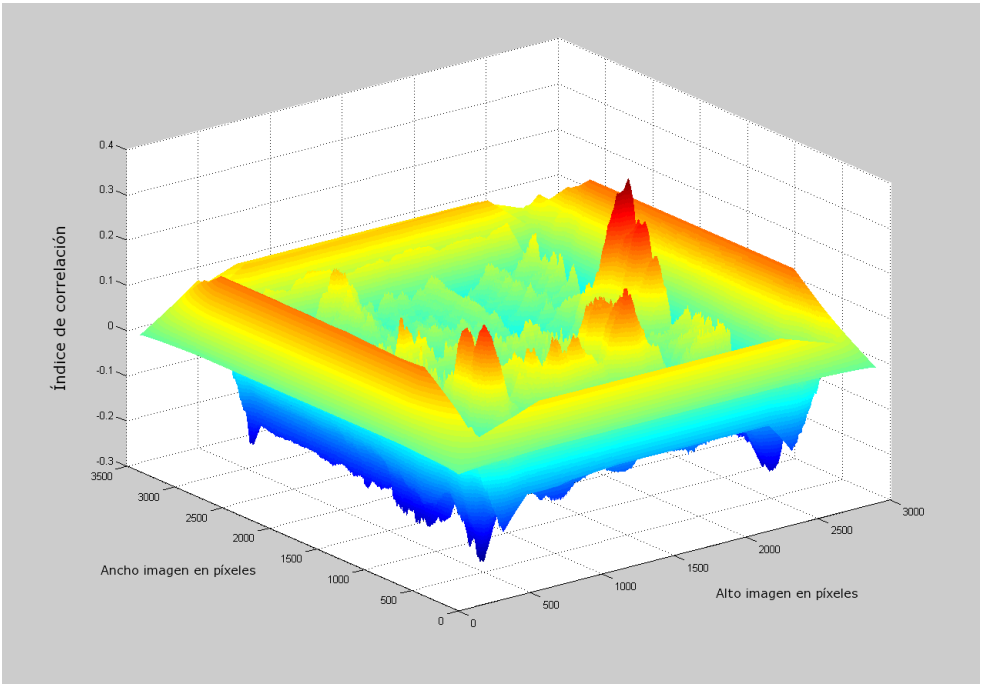

FIGURA 4.6: RESPUESTA POR PÍXEL DE PHASE CORRELATION PARA PÁGINA CON NEGATIVO DEL LOGOTIPO

El coste temporal de estos experimentos es otro gran inconveniente del método *Phase Correlation*. El proceso de búsqueda o *matching* de un logotipo en una página publicitaria se mide en horas. Este tiempo es muy elevado y completamente inviable de cara a una aplicación real. La detección manual o visual de un logotipo por una persona, es muchísimo menor que la realizada por este método, por lo que no tiene sentido utilizar entonces algoritmos que impliquen este coste. Es necesario por lo tanto, construir sistemas que sean capaces de dar una respuesta rápida y fiable en un tiempo adecuado.

4.2 SPEEDED-UP ROBUST FEATURES. SURF

En este apartado se presentan los resultados de toda la experimentación realizada sobre el sistema final con el algoritmo de extracción de características SURF. El capítulo se divide en diferentes bloques correspondientes a las diferentes etapas en el diseño del sistema de reconocimiento de logotipos: Pre-proceso, Estudio de parámetros óptimos, Clasificación y Post-proceso. En cada uno de estos bloques se presentan los resultados obtenidos aplicando los métodos y algoritmos anteriormente explicados para cada apartado. Por último, se presentan resultados de determinados experimentos realizados sobre la plataforma MATLAB® y sobre la librería IRIS implementada en C#.

Para todas las tablas de resultados que se presentan a continuación se seguirá la siguiente notación:

- TP: Verdaderos positivos
- TN: Verdaderos negativos
- FP: Falsos positivos
- FN: Falsos negativos
- TPP: Puntos verdaderos positivos (punto *matcheados* sobre los logotipos)
- FPP: Puntos falsos positivos (puntos *matcheados* fuera de los logotipos)
- tpRate: Ratio de verdaderos positivos
- fpRate: Ratio de falsos positivos
- Sens: Sensibilidad
- Spec: Especificidad
- Prec: Precision
- Acc: Accuracy
- Time: (Tiempo de extracción características + Tiempo *matching*) en segundos
- TFV: Valor de la función objetivo

4.2.1 IMPLEMENTACIÓN DE LA LIBRERÍA FINAL. IRIS

Se ha implementado una librería final de carácter comercial del sistema de reconocimiento de logotipos y marcas para prensa escrita. La librería se llama IRIS y se ha implementado finalmente en lenguaje C# y .NET Framework 4, con el objetivo de integrarla en el sistema de información de inteligencia mediática.

La librería hace uso de la implementación oficial del algoritmo SURF llevada a cabo por *Chris Evans* que se puede encontrar en el siguiente *link*: http://www.chrisevansdev.com/computer-vision-opensurf.html. Se han modificado algunos aspectos de esta implementación, con el objetivo de incrementar el rendimiento y velocidad de la misma. Principalmente se ha modificado la clase *IntegralImage* de forma que se procesa la imagen original mediante código *unsafe*, accediendo de forma no administrada al bloque de memoria en el que se almacena dicha imagen, procesándola así de forma más rápida. Esta optimización incrementa en torno a un 20% la velocidad de extracción de características y mejora por lo tanto las prestaciones del sistema.

IRIS hace uso de la librería *Numerics* del proyecto *MathNet* de procesamiento matemático, cálculo estadístico y algebra lineal. Se hace uso de esta librería para cálculo y operaciones con matrices, y para el desarrollo del modelo discriminante estadístico de la eliminación de falsos positivos. La librería *Numerics* está en continuo desarrollo por el proyecto *MathNet* y está considerada la librería de cálculo numérico *open source* más potente para C#.

La librería IRIS consta actualmente de 665 líneas de código y 15 clases con el siguiente diagrama de dependencia:

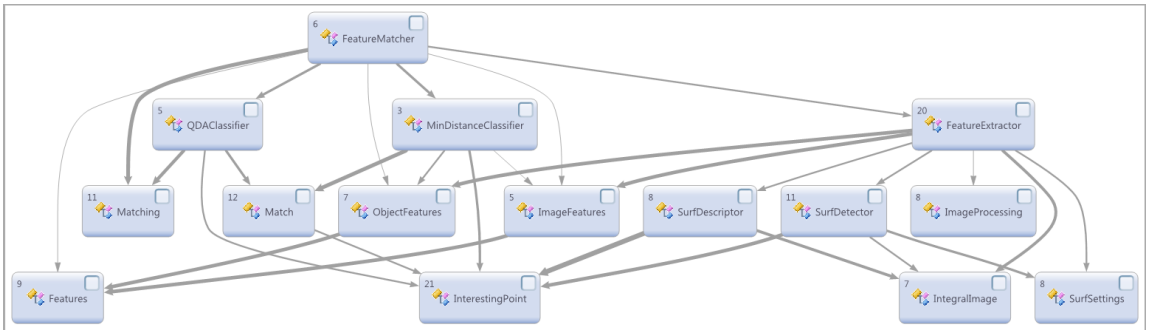

FIGURA 4.7: DIAGRAMA DE DEPENDENCIA DE CLASES DE LA LIBRERIA IRIS

La librería exporta principalmente dos clases públicas como interfaz de uso de la misma. Estas son la clase *FeatureExtractor* y la clase *FeatureMatcher*.

La clase *FeatureExtractor* es la encargada de la extracción de características de las imágenes y por lo tanto del manejo del algoritmo SURF. Esta clase devuelve a través de sus métodos objetos de tipo *ObjectFeatures* e *ImageFeatures* que contienen los puntos interesantes extraídos de las imágenes. La clase *FeatureExtractor* es también la encargada de controlar el pre-proceso que se le aplica a las imágenes. La clase *ImageProcessing* implementa actualmente el pre-proceso de inversión de color del logotipo y se está desarrollando actualmente el ajuste del tamaño del lienzo y la descomposición RGB.

La clase *FeatureMatcher* es la encargada del proceso de *matching* entre características extraídas de las imágenes. Principalmente el método *Match* de esta clase invoca a la clase *MinDistanceClassifier* que toma dos objetos *ObjetctFeatures* e *ImageFeatures* y compara los puntos de uno con el otro en busca de los posibles *matchings*. La clase *FeatureMatcher* es también la encargada de controlar el post-proceso o filtrado de los falsos positivos. La clase *QDAClassifier* implementa el clasificador cuadrático condicho fin. Finalmente se devuelve un objeto *Matching* que consta con una lista de objetos *Match* correspondientes con cada uno de los puntos que se ha conseguido *matchear* y diversa información útil del proceso de *matching*.

La librería está actualmente en pleno desarrollo. Existe una versión de la librería en producción en la empresa *AuditMedia*, para dar servicio de reconocimiento de logotipos en prensa escrita a los clientes.

4.2.2 PRE-PROCESO

En este apartado se presentan los diferentes resultados obtenidos tras aplicar varias de las técnicas propuestas para el pre-proceso. No se incluyen resultados para cada una de las técnicas puesto que varias de estas se descartaron inmediatamente sin necesidad de realizar una experimentación completa.

4.2.2.1 TÉCNICAS SOBRE PÍXEL

4.2.2.1.1 INVERSIÓN DE COLOR

La inversión de color contribuye de forma muy significativa a la mejora de los resultados del sistema. En la siguiente figura se muestra un ejemplo de la necesidad de la inversión del logotipo. Es necesario invertir el color del logotipo para obtener el mismo tipo de contraste que aparece en la página. En este caso, el logotipo de la página presenta elementos claros sobre un fondo oscuro, que se corresponde exactamente con el tipo de contrastes de la imagen invertida en color.

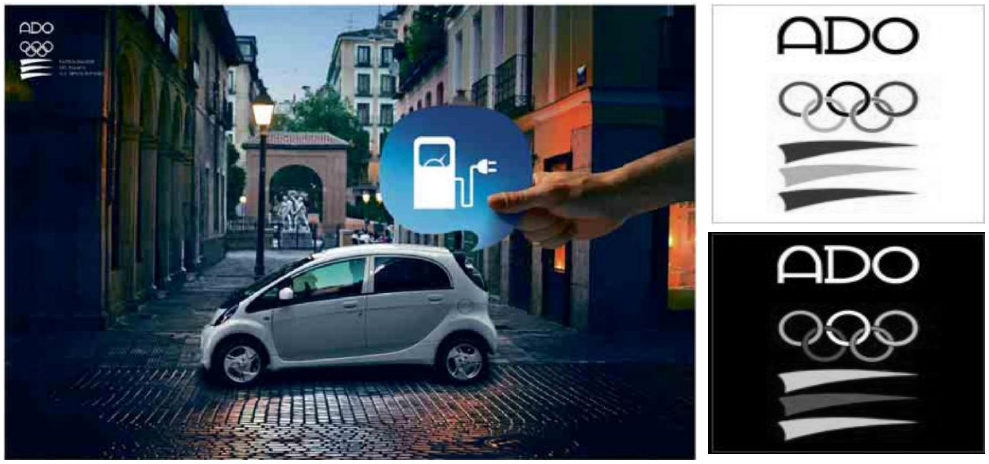

FIGURA 4.8: A LA IZQUIERDA IMAGEN PUBLICITARIA CON LOGOTIPO INVERTIDO. A LA DERECHA, ARRIBA LOGOTIPO EN ESCALA DE GRISES ORIGINAL, ABAJO LOGOTIPO EN ESCALA DE GRISES E INVERTIDO

A continuación, se presenta la comparativa de los resultados del análisis sin aplicar esta técnica y aplicando la inversión de color.

DATOS DEL ANÁLISIS

- Base de datos de páginas publicitarias: *Mono-Medio* (64 páginas)
- Base de datos de logotipos: *Mono-Medio* (22 logotipos)
- Verdaderos positivos: 27
- Verdaderos negativos: $(64 * 22) - 27 = 1381$
- Plataforma: IRIS C#
- Parámetros SURF: Configuración óptima

Los resultados obtenidos por el sistema de reconocimiento de logotipos y marcas son:

	TP	TN	FP	FN	TPP	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc
Con inversión	26	1285	96	1	556	366	0.96	0.07	0.96	0.93	0.21	0.93
Sin inversión	21	1329	52	6	465	194	0.78	0.04	0.78	0.96	0.29	0.96

TABLA 4: RESULTADOS DEL ÁNALISIS DE INVERSIÓN DE COLOR

Como se puede ver en los resultados, la inversión de color del logotipo aporta de forma significativa a la mejora de los resultados. Mediante esta técnica se consiguen detectar todos los casos en los que el logotipo aparecía invertido y en silueta. Por otro lado, es normal que al evaluar los logotipos 2 veces se incremente la tasa de *falsos positivos*. Para finalizar, el *falso negativo* restante se corresponde con un caso de extrema compresión y baja calidad, sobre el que no se ha podido detectar en ninguno de los casos la existencia del mismo.

4.2.2.2 FILTRADO

4.2.2.2.1 FILTRO CANNY

El objetivo del filtro *Canny* consiste en realzar los bordes de los posibles logotipos contenidos en la imagen, de forma que estos se diferencien del resto de elementos de la página. Sin embargo, el filtro *Canny* produce un efecto no deseado para el algoritmo de extracción de características. Los filtros de realce de bordes generan una imagen binarizada de fondo negro, en la que se representan todos los bordes mediante líneas blancas. Esto produce un alto contraste en toda la imagen e implica perder información valiosa como el color o los gradientes. Estos filtros tampoco presentan un buen comportamiento con los fondos con texturas. En la Figura 4.9, el fondo de nubes genera una cantidad de pequeños bordes y de saltos de contraste que introducen muchísimo ruido sobre la imagen. Esto implica que se multiplique exponencialmente el número de puntos interesantes de forma no deseada, ya que son puntos que realmente no describen información relevante de la imagen. Esto también implica que se multiplique más aún el número de comparaciones a realizar y por lo tanto el tiempo de procesamiento de la página.

FIGURA 4.9: DERECHA IMAGEN FILTRO CANNY Y DETALLE. IZQUIERDA IMAGEN ORIGINAL Y DETALLE

En conclusión, se descartó el uso de esta herramienta en el pre-proceso debido a que no introducía ningún beneficio en el sistema.

4.2.2.2.2 FILTRO DENOISING

El filtro *denoising* finalmente no ha mejorado el rendimiento del sistema. La Figura 4.10 representa una aplicación real del filtro *Wiener* para la eliminación de artefactos JPEG. Ambos casos se han evaluado tomando un kernel de 5×5 sobre el píxel afectado. Como se puede observar, se consigue eliminar parte del defecto JPEG, sin embargo al reemplazar el valor de cada pixel de la imagen por la media de los valores de los píxeles cercanos, se produce un *smoothing* sobre la imagen que difumina los contrastes de la misma. Este efecto provoca que los gradientes de la imagen disminuyan e incluso puedan llegar a variar notablemente, llegando a ser diferentes a los de la imagen original.

FIGURA 4.10: EJEMPLO DE APLICACIÓN REAL DEL FILTRO WIENER PARA ELIMINACIÓN DE ARTEFACTOS JPEG

Este es el principal motivo del filtro. En conclusión, el filtro propuesto no introdujo ninguna mejora en el proceso detección de los logotipos, incluso en ocasiones llegó a empeorar los resultados. En determinadas situaciones se detectaron menos puntos correlacionados por *matching*, e incluso menos detecciones completas. Por todos estos motivos, se descartó finalmente el uso de esta técnica en el pre-proceso.

4.2.2.3 GESTIÓN DE LA INFORMACIÓN DE COLOR

En este apartado se presentan los resultados obtenidos por las técnicas *PCA* y *RGB Decomposition*, así como una comparativa de los tres métodos en cuanto a tiempos, costes y resultados.

4.2.2.3.1 PRINCIPAL COMPONENT ANALYSIS

Mediante la técnica PCA se consigue aumentar en la mayoría de casos el número de puntos de *matching* entre logotipos y páginas publicitarias. Incluso se llega a detectar casos en los que con la aproximación de escala de grises no se detecta. En la Figura 4.12 se muestra una proyección de la primera componente de color del logotipo, sobre la página publicitaria y sobre el propio logotipo. En la Figura 4.12 se muestra la imagen y el logotipo originales en escala de grises.

FIGURA 4.11: PROYECCIÓN EN ESCALA DE GRISES DEL COLOR MÁS SIGNIFICATIVO DEL LOGOTIPO, SOBRE LA PÁGINA PUBLICITARIA Y SOBRE EL PROPIO LOGOTIPO

FIGURA 4.12: PÁGINA PUBLICITARIA Y LOGOTIPO ORIGINAL EN ESCALA DE GRISES

Se puede observar la diferencia de contrastes de la proyección con respecto a la imagen original en escala de grises. Incluso pueden verse zonas que desaparecen en la proyección PCA, como el área inferior de la imagen o determinadas zonas del logotipo. Sin embargo el contraste del logotipo se mantiene e incluso se incrementa ligeramente, resaltando por encima del resto.

Aun cumpliendo su propósito, PCA presenta un gran inconveniente, el coste computacional. La cantidad extracción de puntos interesantes que es necesario realizar es muy elevada. Dado un conjunto de N logotipos y M páginas publicitarias, el coste computacional de la extracción de puntos interesantes de ambos conjuntos es $3 \times (2\alpha N + \beta(N \times M))$, con $N \ll M \wedge \alpha, \beta$ los coeficientes del coste de extracción de puntos, mientras que en el proceso original es $2\alpha N + \beta M$. Los coeficientes de extracción de puntos α y β ponderan el coste particular de extraer puntos sobre los logotipos y sobre las páginas publicitarias, con respecto al coste global. Se ha determinado que sobre el cómputo global, la extracción de puntos sobre imágenes implica un 95% del coste, mientras que la extracción en logotipos ocupa el 5% restante. Por lo tanto, los coeficientes α y β toman los valores $\alpha = 0.05$ y $\beta = 0.95$. Por otro lado, en PCA, es necesario proyectar cada uno de los logotipos contra todas las páginas publicitarias, de ahí el $\beta(N \times M)$. Además, se comprobó que no se podía asumir que con la primera componente bastaba para cubrir las necesidades del sistema. Se observaron casos en los que la detección de puntos se correspondía con la proyección de la segunda componente o la tercera, lo que obliga finalmente a calcular todas estas proyecciones de colores predominantes. Por este motivo es necesario repetir el proceso 3 veces, de ahí el $3 \times (\dots)$.

4.2.2.3.2 RGB

Los resultados obtenidos por esta técnica se asemejaron e incluso mejoraron los obtenidos por PCA con un coste computacional muchísimo menor. Evaluar cada capa de color por separado ofrece un análisis de la influencia del color sobre el proceso de *matching* y ofrece determinadas ventajas a tener en cuenta.

FIGURA 4.13: LOGOTIPO DE EJEMPLO DE LA BASE DE DATOS MULTI-MEDIO

FIGURA 4.14: ANÁLISIS RGB DE UNA PÁGINA PUBLICITARIA DE LA BASE DE DATOS MULTI-MEDIO. DE IZQUIERDA A DERECHA: CANAL ROJO, CANAL VERDE Y CANAL AZUL

En la Figura 4.14 se puede observar una aplicación real de la técnica *RGB Decomposition*. El contraste que se produce en el logotipo de ejemplo es diferente en cada canal. En el canal rojo el contraste es el máximo. Esto es debido a que el color principal del logotipo ejemplo es de un azul verdoso por lo que toda la información se sitúa en las capas verde y azul. Por lo tanto dicho logotipo no tiene prácticamente componente roja, y por lo tanto su valor en esta capa es cercano a 0 que en escala de grises es negro. En el resto de capas, la estrella del logotipo aparece prácticamente en blanco, lo que significa que en esos canales se observan valores muy altos cercanos a 255, es decir blanco.

El coste computacional de esta técnica es muchísimo menor que PCA. Es necesario realizar únicamente tres extracciones de puntos, una por cada canal, de cada imagen y logotipo, por lo que el coste computacional se reduce a $3 \times (2\alpha N + \beta M)$, con $N \ll M \wedge \alpha, \beta$ los coeficientes del coste de extracción de puntos, que es lineal y muy inferior que el de PCA.

4.2.2.3.3 Escala de grises vs Principal Component Analysis vs RGB Decomposition

A continuación se presentan los resultados de la comparativa entre las diferentes técnicas de gestión de información de color:

DATOS DEL ANÁLISIS

- Base de datos de páginas publicitarias: *Phantom* (1 página)
- Base de datos de logotipos: *Multi-Medio* (41 logotipos)
- Verdaderos positivos: 41
- Verdaderos negativos: $(1 * 41) - 41 = 0$
- Plataforma: MATLAB®
- Parámetros SURF: *InitSample* = 2, *Octaves* = 4, *Threshold* = 0.0013 (Configuración NO óptima)

NORMAL VS PCA VS RGB DECOMPOSITION

	TP	TN	FP	FN	TPP	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc	TFV
Escala de grises	19	0	8	22	382	40	0.46	-	0.46	-	0.71	0.46	0.72
PCA	24	0	4	17	1099	94	0.58	-	0.58	-	0.86	0.58	0.63
RGB	25	0	3	16	1047	62	0.61	-	0.61	-	0.89	0.61	0.79

TABLA 5: COMPARATIVA DE RESULTADOS DE LAS TÉCNICAS DE GESTIÓN DEL COLOR

	Extracción de características de la DB (s)	Proceso de *matching* (s)
Normal	195	817
PCA	20736	2467
RGB	557	2281

TABLA 6: COMPARATIVA DE TIEMPOS DE LAS TÉCNICAS DE GESTIÓN DE COLOR

En primer lugar, hay que notar que la tasa de detección es relativamente baja. El máximo ratio de detección es de 25 logotipos de los 41 existentes. Esto es normal puesto que el diseño del *Phantom* está orientado a poner a prueba el sistema y dificultar al máximo la tarea de detección. Como se puede ver en la tabla de la comparativa de los resultados, el pre-proceso *RGB Decomposition* es el que mejores prestaciones ofrece. En la gran mayoría de los estimadores calculados, esta técnica es la que mejores valores presenta. Los estimadores *fpRate* y *Specificity* no es posible calcularlos debido a que estamos operando sobre el *Phantom*. En cuanto a la tabla de comparativa de tiempos, los mejores resultados los ofrece obviamente el análisis en escala de grises, sin embargo se puede observar la gran diferencia de aplicar *RGB Decomposition* con respecto a *PCA* en el proceso de extracción de puntos.

4.2.2.4 AJUSTE DEL TAMAÑO DE LIENZO

Actualmente no se dispone de una técnica cerrada para solucionar este problema. Se estima que mediante este pre-proceso se puede llegar a incrementar en torno a un 15% el número de puntos *matcheados* con respecto a logotipos mal ajustados. Este pre-proceso aplicado a los logotipos aporta sustancialmente calidad al sistema y mejora los resultados obtenidos por el mismo. Aumenta el número de puntos detectados por logotipo, incrementando así la probabilidad de conseguir un *matching*. Evita problemas de detección de logotipos relacionados con la pérdida de puntos o descripción incorrecta de la información, y por último ayuda en una cuantía considerable a fortalecer la relevancia de los *matchings* con más puntos *matcheados* por caso. El único inconveniente es el ligero incremento casi despreciable del tiempo medio de extracción de puntos de los logotipos.

4.2.3 ESTIMACIÓN DE LOS PARÁMETROS ÓPTIMOS

En el capítulo 3.2.2.1 se ha presentado los fundamentos del estudio de los parámetros óptimos del algoritmo SURF para la tarea que se desea resolver. En dicho capítulo se presenta un análisis de 1000 configuraciones sobre una base de datos formada por 41 logotipos y el *Phantom*. Para el análisis de los parámetros óptimos, se ha definido una función objetivo capaz de evaluar los resultados de dichas configuraciones devolviendo un *score*, cuyo argumento máximo devolverá la configuración óptima para el sistema. Los resultados del estudio se presentan a continuación:

DATOS DEL ANÁLISIS

- Base de datos de páginas publicitarias: *Phantom* (1 página)
- Base de datos de logotipos: *Multi-Medio* (41 logotipos)
- Verdaderos positivos: 41
- Verdaderos negativos: $(1 * 41) - 41 = 0$

MATLAB®

Los parámetros óptimos obtenidos en la plataforma MATLAB® han sido:

- $InitSample$: 2
- $Octaves$: 4
- $Threshold$: 0.0005
- $Max\ Target\ Function\ value$: 0.6828

Las siguientes gráficas representan la superficie generada por los valores de respuesta de la función objetivo para las 1000 configuraciones evaluadas. Cada gráfica representa una de las 5 *octavas* computadas y en cada gráfica se representa en el eje X los *thresholds*, en el eje Y los *init samples* y en el eje Z el valor de la función objetivo.

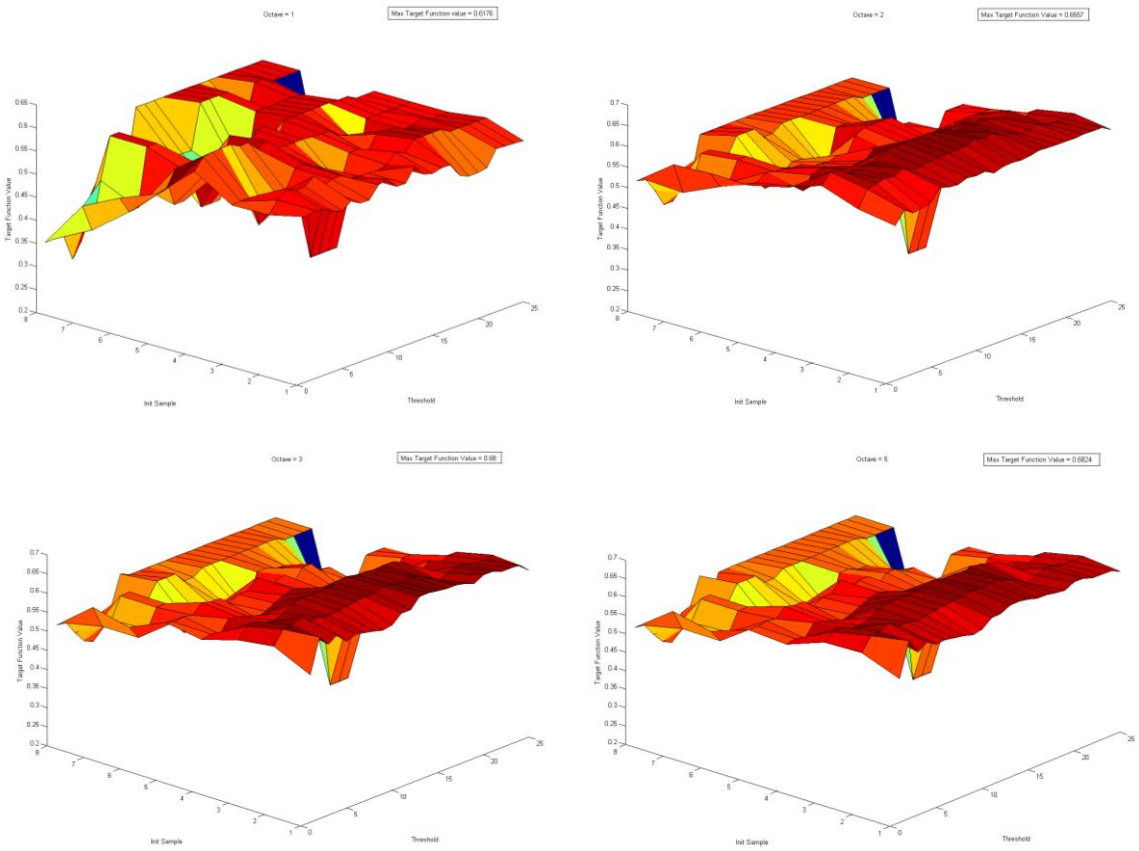

FIGURA 4.15: SURFACE PARÁMETROS ÓPTIMOS MATLAB. OCTAVAS 1, 2, 3 Y 5

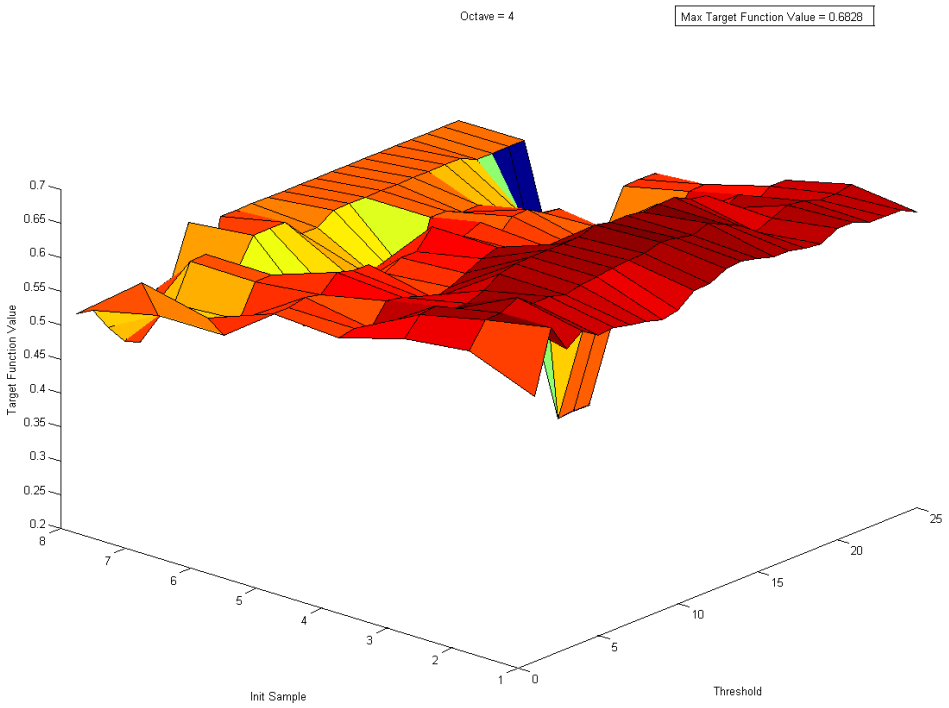

FIGURA 4.16: SURFACE PARÁMETROS ÓPTIMOS MATLAB. CONFIGURACIÓN GANADORA

Notar que el eje de los *initSamples* esta reglado de 1 a 8. Como se comentó en el capítulo 3.2.2.1, se han evaluado 8 *initSamples* comenzando por el valor 2 hasta el 9. Por lo tanto, el valor 1 en la gráfica se corresponde con el *initSample* 2, el valor 3 en la gráfica con el *initSample* 4, etc. De la misma forma, el eje de los *thresholds* comprende 25 valores diferentes correspondientes a los 25 *thresholds* reales que se han evaluado, contenidos en el rango $[0.0005: 0.000416666: 0.01]$.

IRIS (C#)

Los parámetros óptimos obtenidos para la librería IRIS en C# han sido:

- $InitSample: 2$
- $Octaves: 3$
- $Threshold: 0.0022$
- $Max\ Target\ Function\ value: 0.7362$

De igual forma que para la plataforma MATLAB®, las siguientes gráficas representan la superficie generada por los valores de respuesta de la función objetivo para las 1000 configuraciones probadas. Cada gráfica representa una de las 5 *octavas* evaluadas y en cada gráfica se representa en el eje X los *thresholds*, en el eje Y los *init samples* y en el eje Z el valor de la función objetivo.

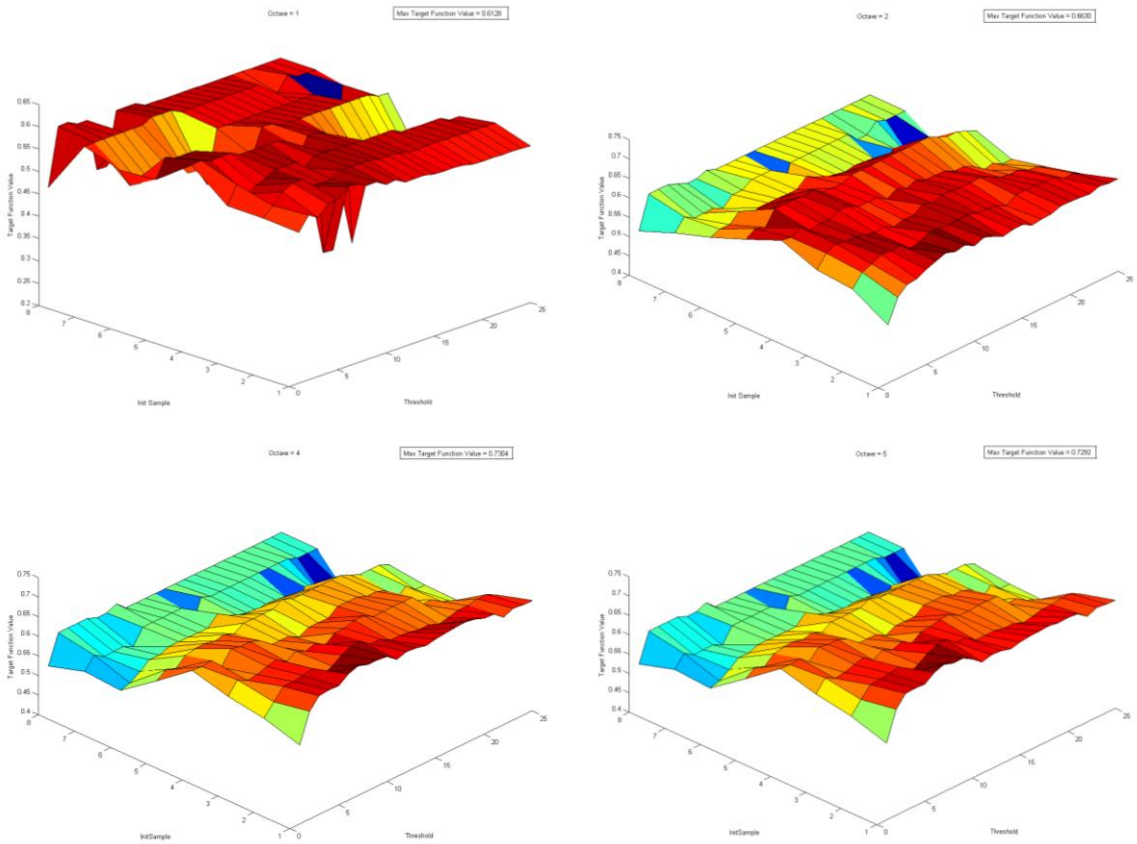

FIGURA 4.17: SURFACE PARÁMETROS ÓPTIMOS IRIS. OCTAVAS 1, 2, 4 Y 5

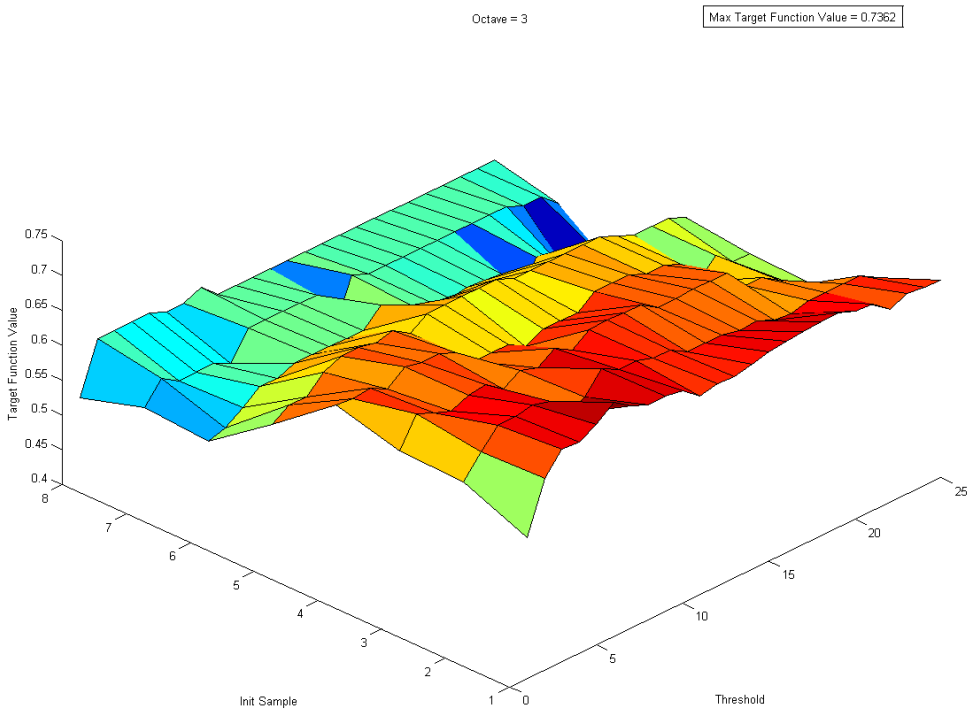

FIGURA 4.18: SURFACE PARÁMETROS ÓPTIMOS IRIS. CONFIGURACIÓN GANADORA

En primer lugar, las configuraciones óptimas para ambas plataformas difieren en cierta medida. Esto nos indica que la implementación en C# es más potente y de mejor calidad puesto que con parámetros de menor exigencia, obtiene resultados similares o mejores que MATLAB®. Por otro lado, queda de manifiesto que la librería IRIS implementada en C# ofrece una mejor respuesta que la implementación en MATLAB®. Se observa una diferencia notable en el valor máximo de la función objetivo, por lo que en base a este estimador, C# se comporta mejor que MATLAB®. En principio podría atribuirse esta mejora únicamente a la reducción del tiempo de cómputo, sin embargo el análisis de los resultados para ambas configuraciones óptimas determinan que en cualquiera de los casos, la implementación del sistema en C# es mejor.

	TP	TN	FP	FN	TPP	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc	TFV	Time (s)
IRIS C#	23	0	3	18	431	23	0.56	-	0.56	-	0.88	0.56	0.73	179
MATLAB	23	0	4	18	436	32	0.56	-	0.56	-	0.85	0.56	0.68	4536

Por otro lado, si analizamos las superficies generadas, podemos observar que, a pesar de que siguen un comportamiento común, la repuesta de la función objetivo para la implementación en C# es más regular que en MATLAB®. No se observan tantos cambios bruscos y si una tendencia a que si se incrementa tanto el *threshold* como los *initSamples*, los resultados empeoran de forma suave.

En conclusión, a partir de estos resultados hemos determinado los parámetros óptimos para la tarea que se desea resolver, tanto para la plataforma MATLAB® como para IRIS. Por otro lado, hemos obtenido una comparativa entre ambas implementaciones, útil para estimar el grado de fiabilidad de las mismas y finalmente poder elegir la mejor de ambas.

4.2.4 Clasificación (Matching)

4.2.4.1 Estrategia Voraz

Los resultados arrojados por la estrategia voraz mejoran sustancialmente los resultados generados por el sistema. Para evaluar las prestaciones de la optimización, se ha realizado el siguiente análisis.

DATOS DEL ANÁLISIS

- Base de datos de páginas publicitarias: *Multi-Medio* (40 página)
- Base de datos de logotipos: *Multi-Medio* (41 logotipos)
- Verdaderos positivos: 53
- Verdaderos negativos: $(40 * 41) - 53 = 1587$
- Plataforma: IRIS C#
- Sin eliminación de *falsos positivos*

La comparativa de los resultados con la aplicación de la estrategia y sin aplicarla:

	TP	TN	FP	FN	tpRate	fpRate	Sens	Spec	Prec	Acc
SIN Estrategia Voraz	52	1387	200	1	0.9811	0.126	0.9811	0.8740	0.2063	0.8774
CON Estrategia Voraz	52	1440	147	1	0.9811	0.0926	0.9811	0.9074	0.2613	0.9098

TABLA 7: COMPARATIVA DE RESULTADOS DE LA OPTIMIZACIÓN POR ESTRATEGIA VORAZ

La reducción de *falsos positivos* es notable. Se consigue reducir en torno a un 27% el número de *falsos positivos* manteniendo exactamente los mismos *verdaderos positivos*, sin perder en ninguno de

ellos ningún punto *matcheado* correctamente. Queda de manifiesto que la optimización a través de esta estrategia aporta calidad al sistema y mejora los resultados obtenidos por el mismo.

4.2.5 CLASIFICACIÓN (POST-PROCESO)

En este apartado se presentan las variables elegidas para el modelo discriminante de clasificación y la validación del mismo para la eliminación de los *falsos positivos*.

4.2.5.1 SELECCIÓN Y ENTRENAMIENTO DE VARIABLES DISCRIMINANTES DEL MODELO

Las variables finales que se han estudiado para construir el modelo final de clasificación han sido: La cantidad de puntos del *matching*, el IQR de la distribución del ratio de distancias, la diferencia entre la media y la mediana de dicha distribución y el estimador MAD también de esta distribución. La siguiente figura muestra todas las combinaciones posibles, dos a dos, de estas variables sobre los resultados obtenidos en el análisis de la base de datos *Multi-Medio*.

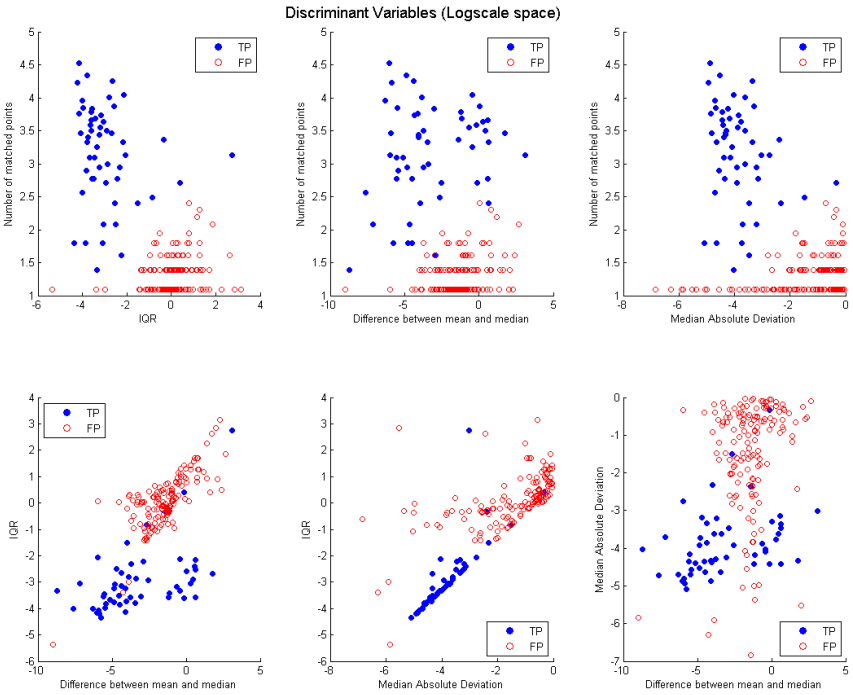

FIGURA 4.19: MODELOS DE CLASIFICACIÓN. COMPARACIÓN DE VARIABLES DISCRIMINANTES. ESCALA LOGARÍTMICA. DISTRIBUCIONES DEL RATIO DE DISTANCIAS NORMALIZADAS CON RESPECTO A SU MEDIANA.

Notar que todas las gráficas están representadas en escala logarítmica. Por otro lado, las distribuciones del ratio de distancias han sido normalizadas dividiendo cada de ellas una entre su propia mediana, por lo que las variables derivadas de esta distribución (*IQR, diferencia entre media y mediana, MAD*) se ven afectadas por esta normalización.

En cada una de las sub-gráficas que se muestran en la Figura 4.19 se representan 199 puntos, correspondientes a los 199 resultados obtenidos tras aplicar la clasificación propuesta en el capítulo 3.2.3 con la estrategia voraz a la base de datos y conjunto de logotipos mencionado anteriormente. Estos puntos se separan en dos clases representadas por los puntos azules y rojos. Los puntos azules se corresponden con los *verdaderos positivos* mientras que los puntos rojos son *falsos positivos*. Como se puede ver, la mayoría de modelos discriminantes presentan unas condiciones de clasificación similares. Uno de los requisitos a tener en cuenta es que el sistema debe ser de alta *sensibilidad*, es decir debe

priorizar la detección de *verdaderos positivos*. Si analizamos los resultados obtenidos, los 199 *matchings* se reparten en 52 *verdaderos positivos* y 147 *falsos positivos*. El objetivo de esta clasificación es filtrar el máximo de *falsos positivos* posibles sin clasificar ningún *verdadero positivo* de forma incorrecta.

En conclusión, el modelo final elegido se corresponde con el primer gráfico propuesto, en el que las dos variables que formarán dicho modelo son: el número de puntos correlacionados en el *matching* y el IQR de la distribución del ratio de las distancias entre los puntos.

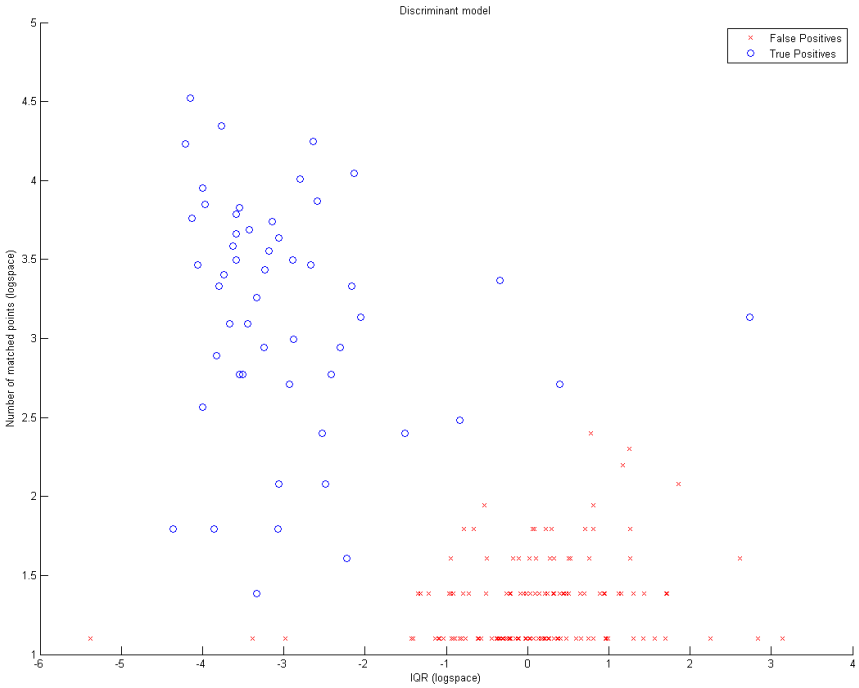

FIGURA 4.20: MODELO DISCRIMINANTE FINAL PARA VERDADEROS POSITIVOS Y FALSOS POSITIVOS

Una vez determinado el modelo discriminante, queda entrenar un clasificador con dicho modelo y testearlo para evaluar su rendimiento.

Tras evaluar los diferentes tipos de clasificadores presentados en el capítulo 3.2.4.2, se eligió el clasificador de *mahalanobis* por ofrecer las mejores condiciones de clasificación en cuanto a separabilidad de las clases. En consecuencia, se ha entrenado el clasificador a partir de las 199 muestras obtenidas del análisis de la base de datos *Multi-Medio.* En la Figura 4.21 se muestran las regiones y fronteras de decisión entre ambas clases. Cada triángulo azul o rojo de la gráfica se corresponde con un *verdadero positivo* o un *falso positivo* respectivamente del conjunto de entrenamiento.

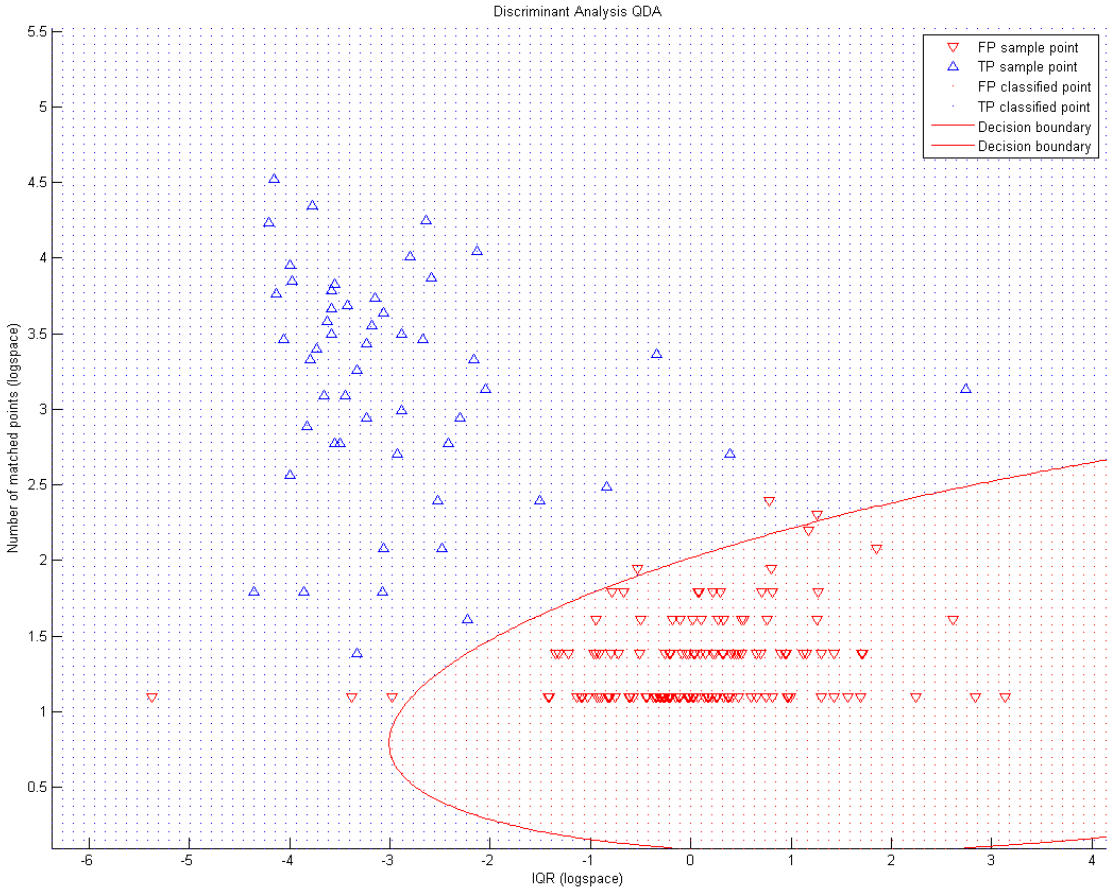

FIGURA 4.21: ANÁLISIS DISCRIMINANTE. FRONTERA DE DECISIÓN

Los parámetros obtenidos del clasificador son:

$$W_{TP} = \begin{bmatrix} 0.1723 & -0.5132 \\ -0.5132 & 5.8571 \end{bmatrix} \qquad W_{FP} = \begin{bmatrix} -0.1723 & 0.5132 \\ 0.5132 & -5.8571 \end{bmatrix}$$

$$w_{TP} = \begin{bmatrix} -0.3332 \\ -12.3654 \end{bmatrix} \qquad w_{TP} = \begin{bmatrix} 0.3332 \\ 12.3654 \end{bmatrix}$$

$$w_{TP0} = 1.1108 \qquad w_{TP0} = -1.1108$$

Se puede observar que los parámetros para la clase de los *verdaderos positivos* son los mismos que los de la clase *falsos positivos* pero invertidos. Esto es normal puesto que para dos clases sólo existe una única frontera de decisión. Un clasificador de 2 clases se puede modelar como un clasificador de una sola clase (*One-class classifier*) en la que, en vez de decidir a cuál de las dos clases pertenece una muestra, la clasificación consiste en decidir si la muestra pertenece o no a la clase modelada.

En Figura 4.21 podemos observar varias muestras mal clasificadas, únicamente del conjunto de los *falsos positivos*. En este caso cumplimos perfectamente el requisito de no clasificar incorrectamente ningún *verdadero positivo*.

A continuación se muestra una tabla con toda la información sobre el modelo discriminante y la clasificación realizada de 35 resultados representativos. Notar que en esta tabla no se ha aplicado el logaritmo a los valores de IQR ni al número de puntos *matcheados*.

TABLA 8: VALORES DEL MODELO DISCRIMINANTE

	Página publicitaria	Logotipo	*Nº matched points*	*IQR*	*TP*	FP
1	Image 001	ADO	6	0.046315	1	0
2	Image 001	Barcena	3	0.894010	0	1
3	Image 001	Endesa	3	0.442446	0	1
4	Image 001	La Caixa	42	0.042990	1	0
5	Image 001	Rolex	3	0.546037	0	1
6	Image 001	Royal Caribbean	3	0.436068	0	1
7	Image 001	Sol Meliá	6	0.454625	0	1
8	Image 002	Banco Santander	3	0.706738	0	1
9	Image 002	Halcón Viajes	4	1.241566	0	1
10	Image 002	JMJ	4	1.570814	0	1
11	Image 002	MINI	20	0.056138	1	0
12	Image 002	Royal Caribbean	3	0.893388	0	1
13	Image 002	Spanair	4	0.602019	0	1
14	Image 003	El Corte Inglés	40	0.032565	1	0
15	Image 003	Halcón Viajes	4	1.158367	0	1
16	Image 003	JMJ	3	0.399469	0	1
17	Image 003	Sol Meliá	16	0.028787	1	0
18	Image 004	El Corte Inglés	52	0.018270	1	0
19	Image 004	Halcón Viajes	4	0.380309	0	1
20	Image 004	JMJ	47	0.018815	1	0
	…	…	…	…	…	…
21	Image 007	Richard Mille	15	1.484329	1	0
22	Image 007	Rolex	3	1.816368	0	1
23	Image 007	Royal Caribbean	5	0.833398	0	1
24	Image 008	El Corte Inglés	69	0.014866	1	0
25	Image 008	JMJ	33	0.027634	1	0
26	Image 009	Ayuntamiento Valencia	4	0.812673	0	1
27	Image 009	Catai	19	0.039196	1	0
28	Image 009	JMJ	3	4.781511	0	1
29	Image 009	Richard Mille	3	0.542340	0	1
30	Image 010	Hofmann	70	0.071322	1	0
	…	…	…	…	…	…
31	Image 014	El Corte Inglés	77	0.023007	1	0
32	Image 014	Ibero Cruceros	29	0.714662	1	0
33	Image 014	JMJ	6	1.070279	0	1
34	Image 014	Richard Mille	3	3.672880	0	1
35	Image 014	Royal Caribbean	3	1.441387	0	1
	…	…	…	…	…	…

En estas 35 detecciones se encuentran 14 *verdaderos positivos* de los 52 existentes y 21 *falsos positivos* de los 147. Como se observa, en la mayoría de los *verdaderos positivos* se consigue tanto un alto número de puntos correlacionados, como un valor muy bajo de IQR, del orden de $0.0xxx$. Sin embargo existen *casos límite* en ambos sentidos. Si se observa la fila 21, se puede ver una detección del logotipo *Richard Mille* en la página 7, en la que su valor de IQR esta incluso por encima de 1. Existen *falsos positivos* con valores de IQR mucho menores por lo que fijándonos únicamente en este valor daríamos por *falso* este resultado. Este es un claro ejemplo en el que es necesario combinar el IQR con el número de puntos *matcheados*. Para este caso, el número de puntos *matcheados* asciende a 15, lo que ofrece una seguridad bastante alta de que se trata de un *verdadero positivo*. En la fila 32 también observamos otro caso en el que el IQR es cercano a 1 y entra dentro de los valores que suelen tomar los *falsos* positivos. Este caso consta de 29 puntos *matcheados* y se debe a una traslación de un elemento del logotipo comparado con el encontrado en la página publicitaria. Esta diferencia distorsiona la

distribución del ratio de distancias y incrementa consecuentemente el IQR. En la Figura 4.22 se muestra el caso concreto que arrojó estos valores.

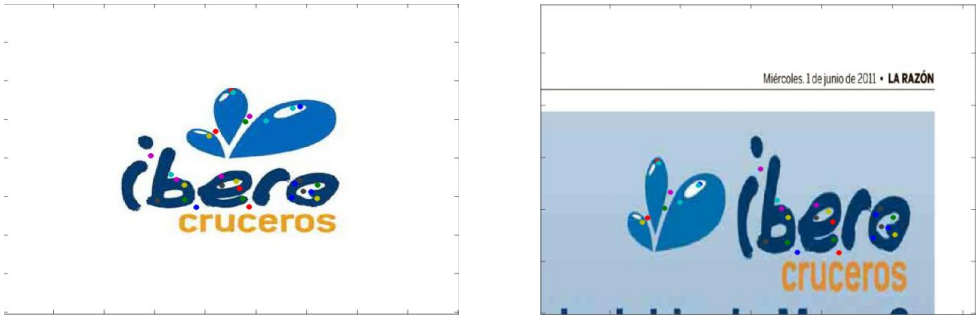

FIGURA 4.22: ELEVADO IQR EN VERDADERO POSITIVO

A priori se puede estimar observando la frontera de decisión del modelo, que se puede conseguir una reducción de *falsos positivos* del $1 - \frac{6}{147} = 0.9592 \approx 96\%$. Esta reducción es muy optimista puesto que se estima a partir de los mismos datos de entrenamiento. Queda entonces validar el modelo para observar su comportamiento en un test independiente.

4.2.5.2 VALIDACIÓN DEL MODELO DISCRIMINANTE

Una vez obtenido el modelo discriminante, es necesario realizar una validación para evaluar el comportamiento del mismo. Para ello se ha hecho uso de la base de datos *Test Independiente*, puesto que está formada por muestras completamente desconocidas para el sistema.

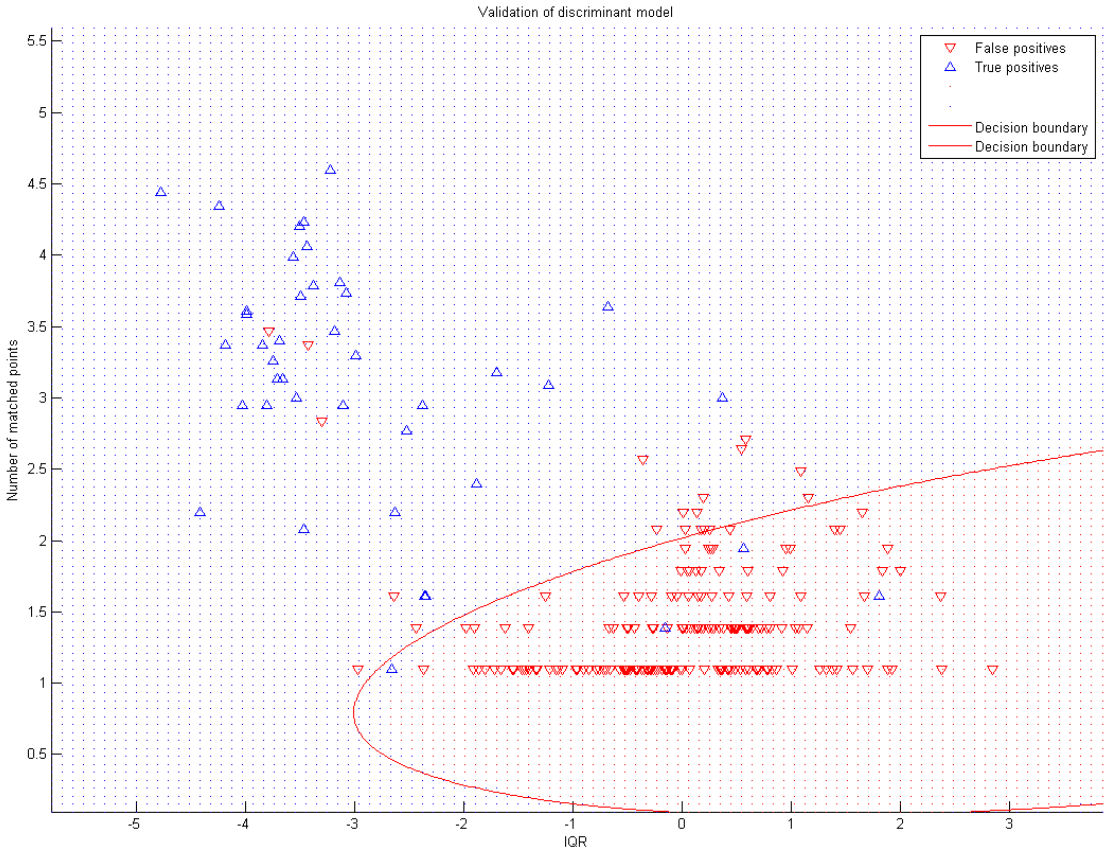

FIGURA 4.23: VALIDACIÓN DEL MODELO DISCRIMINANTE CONTRA LA BASE DE DATOS TEST INDEPENDIENTE

La Figura 4.23 muestra la aplicación del modelo discriminante cuadrático entrenado a partir de los resultados obtenidos de la base de datos *Multi-Medio*, aplicados a los resultados obtenidos de la base de datos *Test Independiente*. Como se puede observar, el modelo se ajusta con bastante precisión a los resultados que cabe esperar. El factor que más importa de este análisis, es la clasificación errónea de los 4 *verdaderos positivos*. Comenzando por el *verdadero positivo* cuyo IQR es el menor, el resultado se corresponde con una detección límite de sólo 3 puntos que sí sigue una distribución de distancias correcta. Este caso se encuentra muy cercano a la frontera de decisión y es de esperar que, tras el refinamiento de la frontera de decisión entrenando el modelo con todas las muestras existentes, dicho caso se corrija pasando a formar parte del conjunto de los *verdaderos positivos*. Por lo que respecta a los 3 casos restantes, ordenados de menor IQR a mayor, dichas detecciones constan de 4, 7 y 5 puntos respectivamente. Todos estos casos presentan unas condiciones de detección en las que varios de los puntos caen fuera del logotipo en la página publicitaria. Es por esto que la distribución de distancias de los puntos, no se encuentra centrada en un único valor, y por lo tanto se dispersa mucho. Por este motivo, dichos casos caen en la región de la clase *Falsos Positivos* y a pesar de que son conocidamente *verdaderos positivos*, es correcto que se sitúen en dicha región puesto que cumplen las condiciones para formar parte de este conjunto. Por otro lado, tras analizar estos casos, se observa rápidamente que las condiciones de resolución y calidad del logotipo en la página, están muy deterioradas.

Por lo que respecta a los *falsos positivos*, el modelo discriminante construido cumple los objetivos marcados, eliminando la gran mayoría de estos casos. Tras el proceso de clasificación, se pasa de tener un total de 200 *falsos positivos,* a únicamente 19 casos. La reducción, como ya se anticipó anteriormente, ronda el 92% de reducción de *falsos positivos*.

4.2.5.3 COMPARATIVA DE RESULTADOS. QDA VS NO QDA

En este apartado se presentan la comparativa de los resultados tras analizar las bases de datos aplicando el clasificador *QDA* para la eliminación de *falsos positivos*, y sin aplicar dicho post-proceso. Los resultados se presentan sobre las plataformas MATLAB® e IRIS, en sus configuraciones óptimas.

BASE DE DATOS MONO-MEDIO (ENTRENAMIENTO)

		TP	TN	FP	FN	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc
MATLAB®	SIN QDA	26	1167	214	1	869	0.96	0.15	0.96	0.84	0.11	0.85
	QDA	26	1373	8	1	83	0.96	0.006	0.96	0.99	0.76	0.99
IRIS	SIN QDA	26	1285	96	1	366	0.96	0.070	0.96	0.93	0.21	0.93
	QDA	26	1378	3	1	15	0.96	0.002	0.96	0.99	0.89	0.99

TABLA 9: COMPARATIVA DE RESULTADOS DE ELIMINACIÓN DE FALSOS POSITIVOS CON EL CLASIFICADOR QDA PARA LA BASE DE DATOS MONO-MEDIO

BASE DE DATOS MULTI-MEDIO (ENTRENAMIENTO)

		TP	TN	FP	FN	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc
MATLAB®	SIN QDA	52	1305	282	1	1221	0.98	0.18	0.98	0.82	0.15	0.83
	QDA	52	1573	14	1	169	0.98	0.009	0.98	0.99	0.79	0.99
IRIS	SIN QDA	52	1440	147	1	592	0.98	0.092	0.98	0.91	0.26	0.91
	QDA	52	1581	6	1	61	0.98	0.004	0.98	0.99	0.89	0.99

TABLA 10: COMPARATIVA DE RESULTADOS DE ELIMINACIÓN DE FALSOS POSITIVOS CON EL CLASIFICADOR QDA PARA LA BASE DE DATOS MULTI-MEDIO

BASE DE DATOS TEST INDEPENDIENTE (TEST)

		TP	TN	FP	FN	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc
IRIS	SIN QDA	42	1205	200	3	959	0.93	0.14	0.93	0.86	0.17	0.86
	QDA	38	1386	19	7	247	0.84	0.01	0.84	0.98	0.67	0.98

TABLA 11: COMPARATIVA DE RESULTADOS DE ELIMINACIÓN DE FALSOS POSITIVOS CON EL CLASIFICADOR QDA PARA LA BASE DE DATOS TEST INDEPENDIENTE

Como se puede ver, el clasificador *QDA* realiza un gran aporte a la calidad del sistema. La reducción de *falsos positivos* alcanza cotas del 92%, manteniendo en la mayoría de los casos el número de *verdaderos positivos* intacto, excepto en el test independiente, que como era de esperar, se clasifican incorrectamente algunas muestras. Observando los estimadores *sensibilidad, especificidad, precisión* y *accuracy*, en la gran mayoría de casos mejoran sus valores incrementando la fiabilidad, robustez y calidad del sistema.

4.2.6 ANÁLISIS DE BASES DE DATOS

En este apartado se presentan los resultados finales obtenidos sobre las diferentes bases de datos, aplicando las siguientes técnicas en cada una de las etapas del sistema:

- Pre-proceso: Inversión de logotipo. Ajuste del lienzo.
- Extracción de características: SURF configuración óptima.
- Matching: Mínima distancia. Estrategia voraz.
- Clasificación: Análisis discriminante cuadrático para *falsos positivos.*

4.2.6.1 BASE DE DATOS MONO-MEDIO

Los resultados que se presentan a continuación se corresponden con el análisis de la base de datos *Mono-Medio* para las plataformas MATLAB® e IRIS con sus configuraciones óptimas. El experimento consta de las siguientes características:

DATOS DEL ANÁLISIS

- Base de datos de páginas publicitarias: *Mono-Medio* (64 página)
- Base de datos de logotipos: *Mono-Medio* (22 logotipos)
- Verdaderos positivos: 27
- Verdaderos negativos: $(64 * 22) - 27 = 1381$
- Plataforma: MATLAB® e IRIS
- Parámetros SURF: Configuraciones óptimas para cada plataforma

Los resultados obtenidos por el sistema de reconocimiento de logotipos y marcas son:

	TP	TN	FP	FN	TPP	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc
IRIS C#	26	1378	3	1	556	15	0.96	0.002	0.96	0.998	0.89	0.997
MATLAB	26	1373	8	1	702	83	0.96	0.006	0.96	0.994	0.76	0.993

TABLA 12: COMPARATIVA DE RESULTADOS ENTRE LA PLATAFORMA MATLAB Y LA LIBRERIA IRIS PARA LA BASE DE DATOS MONO-MEDIO

En primer lugar hay que observar que como se trata de la base de datos Mono-Medio que ha participado en el proceso de entrenamiento, los resultados son muy optimistas y pueden estar sobre ajustados. Como ya se ha visto anteriormente, la implementación en C# mejora a la implementación en MATLAB®. En este caso, se detecta el mismo número de *verdaderos positivos* en ambas plataformas, sin embargo no ocurre lo mismo con los *falsos positivos*, ni con la relación entre la cantidad de puntos

matcheados correctamente e incorrectamente. Los estimadores *sensibilidad, especificidad, precisión* y *accuracy* también varían inclinándose por IRIS como mejor sistema de reconocimiento de logotipos. Por otro lado, el tiempo necesario para obtener estos resultados en IRIS es también mucho menor que en MATLAB®.

4.2.6.2 BASE DE DATOS MULTI-MEDIO

Los resultados que se presentan a continuación se corresponden con el análisis de la base de datos *Multi-Medio* para las plataformas MATLAB® e IRIS con sus configuraciones óptimas. El experimento consta de las siguientes características:

DATOS DEL ANÁLISIS

- Base de datos de páginas publicitarias: *Multi-Medio* (40 página)
- Base de datos de logotipos: *Multi-Medio* (41 logotipos)
- Verdaderos positivos: 53
- Verdaderos negativos: $(40 * 41) - 53 = 1587$
- Plataforma: MATLAB® e IRIS
- Parámetros SURF: Configuraciones óptimas para cada plataforma

Los resultados obtenidos por el sistema de reconocimiento de logotipos y marcas son:

	TP	TN	FP	FN	TPP	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc
IRIS C#	52	1581	6	1	1542	61	0.98	0.004	0.98	0.996	0.89	0.995
MATLAB	52	1573	14	1	1910	169	0.98	0.009	0.98	0.991	0.79	0.991

TABLA 13: COMPARATIVA DE RESULTADOS ENTRE LA PLATAFORMA MATLAB Y LA LIBRERIA IRIS PARA LA BASE DE DATOS MULTI-MEDIO

De la misma forma que en la anterior base de datos, esta base de datos también ha participado en el proceso de entrenamiento, y por lo tanto los resultados son muy optimistas y pueden estar sobre ajustados Repitiendo el comportamiento anterior, la implementación en C# mejora, esta vez con más diferencia, a la implementación en MATLAB®. Igualmente, se detecta el mismo número de *verdaderos positivos* en ambas plataformas, sin embargo otra vez los *falsos positivos* aumentan en MATLAB® con respecto a IRIS. La relación entre la cantidad de puntos *matcheados* correctamente e incorrectamente continúa decantándose por IRIS, y de igual forma los estimadores *sensibilidad, especificidad, precisión* y *accuracy* que también reflejan la ventaja de IRIS con respecto a MATLAB®.

4.2.6.3 BASE DE DATOS TEST INDEPENDIENTE

Los resultados que se presentan a continuación se corresponden con el análisis de la base de datos *Test Independiente* para la plataforma IRIS en su configuración óptima. El experimento consta de las siguientes características:

DATOS DEL ANÁLISIS

- Base de datos de páginas publicitarias: *Test Independiente* (50 página)
- Base de datos de logotipos: *Test Independiente* (29 logotipos)
- Verdaderos positivos: 45
- Verdaderos negativos: $(50 * 29) - 45 = 1405$
- Plataforma: IRIS
- Parámetros SURF: Configuración óptima

Los resultados obtenidos por el sistema de reconocimiento de logotipos y marcas son:

	TP	TN	FP	FN	TPP	FPP	tpRate	fpRate	Sens	Spec	Prec	Acc
IRIS C#	38	1386	19	7	1216	247	0.84	0.013	0.84	0.98	0.67	0.98

TABLA 14: RESULTADOS DE LA LIBRERIA IRIS PARA LA BASE DE DATOS TEST INDEPENDIENTE

Como era de esperar, al evaluar una base de datos independiente, los resultados tienden a empeorar, siendo estos más realistas. Los 7 *falsos negativos* se corresponden con 3 casos que ha sido imposible detectar y con 4 casos que el clasificador *QDA* ha clasificado incorrectamente y por lo tanto ha eliminado. Los 3 casos que no ha sido posible detectar se corresponden principalmente con problemas de diferencia de contraste, es decir, con casos en los que el logotipo que se encuentra en la página no guarda el mismo tipo de contraste que el original y no ha sido posible corregirlo mediante la inversión de logotipo. Sin embargo, este problema se puede resolver aplicando la técnica *RGB Decomposition*.

5 DISCUSIÓN

En este proyecto se presenta un sistema de detección de logotipos y marcas para imágenes estáticas de prensa escrita. El objetivo consiste en construir un sistema de altas prestaciones para la ayuda a la evaluación del impacto publicitario de una entidad, sobre diferentes medios de comunicación.

En primera instancia se abordó la solución al problema mediante el algoritmo *Phase Correlation*. Como ya se ha explicado a lo largo del proyecto, este algoritmo presenta fuertes inconvenientes como la dependencia con respecto a escala, rotación y traslación, la velocidad y su poca capacidad de descripción de las imágenes, que implica partir de un algoritmo no óptimo para resolver la tarea. Por estos motivos, fue necesario buscar otros métodos que se ajustaran mejor a las necesidades del problema. SURF es finalmente el algoritmo elegido que forma el núcleo principal del sistema de reconocimiento de logotipos y marcas.

Para el diseño del sistema, ha sido necesario abordar cada uno de los problemas descritos durante el transcurso del proyecto. Una de las dificultades a las que se enfrenta el sistema reside en la naturaleza propia de las imágenes. Las imágenes digitales *rasterizadas* presentan problemas relacionados con la calidad, compresión, resolución, etc. que pueden dificultar en gran medida la detección de objetos hasta el punto de hacer imposible el reconocimiento de estos. Nuestro sistema de reconocimiento de logotipos distingue entre dos conjuntos de imágenes: las imágenes correspondientes a las páginas publicitarias y las imágenes que representan los logotipos a detectar. Estas imágenes, generalmente son de diferente tamaño y características. Las imágenes de los logotipos son normalmente mucho más pequeñas que las páginas publicitarias, y además son de contenido más sencillo. Generalmente, los logotipos suelen representar una figura más un determinado texto, normalmente con una tipografía no muy común. Sin embargo, las páginas publicitarias suelen estar más recargadas de textos, letreros, fotografías, anuncios e infinidad de objetos, que multiplican la información existente en las mismas. Es por esto, que es altamente preferible aplicar pre-procesos o tratamientos a los logotipos antes que a las páginas publicitarias. El coste de extraer puntos interesantes de una página publicitaria con respecto a un logotipo es en torno a 20 veces mayor, por lo que es obvio que los pre-procesos que impliquen recalcular puntos deben estar orientados preferiblemente a los logotipos.

La técnica de inversión de color del logotipo es una de ellas. Esta técnica es imprescindible para alcanzar los objetivos marcados y asegurar el buen funcionamiento del sistema. Es muy común encontrar en publicidad logotipos en silueta e invertidos con respecto a su color original, y es por lo tanto necesario dar una solución a este problema. En el capítulo 4.2.2 se ha presentado un apartado con el impacto que produce esta técnica sobre los resultados, y se demuestra que su aplicación soluciona al completo el problema planteado. Como se puede ver en estos resultados, sin la inversión de color estamos perdiendo hasta 5 detecciones con respecto al análisis con inversión de color. Por otro lado, es obvio que al evaluar los logotipos dos veces variando el tipo de contraste, se generen más falsos positivos. Sin embargo, los requisitos funcionales del proyecto abogan por un sistema de alta *sensibilidad* por lo que es prioritaria la detección de los máximos logotipos posibles. Si observamos los *sensibilidad*, *especificidad, precisión* y *accuracy*, observamos que la *sensibilidad* mejora notablemente, mientras que la *especificidad, precisión y accuray* empeoran ligeramente en el análisis con inversión de color. Esto es debido obviamente al incremento de los *falsos positivos*, sin embargo este empeoramiento es muy leve con respecto a la mejora que se obtiene en *sensibilidad*, siendo además este último, el valor de mayor importancia. Junto con todo esto, el coste de aplicar dicha técnica no es elevado. La inversión de color del logotipo implica calcular dos veces los puntos interesantes del

logotipo. Se duplica por lo tanto el coste de *matching* entre logotipo e imagen elevándolo a 1 segundo, sin embargo este coste es asumible y necesario para cubrir las necesidades del problema.

En cuanto a los filtros de realce de bordes y de eliminación de ruido, se ha visto que su funcionamiento no aporta ninguna mejora al sistema. Los filtros de realce de bordes generan muchísimo ruido y eliminan información valiosa como el color o los degradados y por lo tanto empeoran notablemente los resultados. Además el propio algoritmo SURF lleva implícito en la extracción de puntos interesantes, una detección de bordes y zonas de alto contraste por lo que se hace innecesaria dicha técnica en el pre-proceso. En cuanto al filtro *Wiener* para la eliminación de artefactos JPEG, tampoco se ha conseguido mejorar el sistema. Este filtro generalmente no empeoraba los resultados pero tampoco introducía ninguna mejora. Dicho filtro se debe aplicar únicamente a las páginas publicitarias puesto que los logotipos se encuentran en buena calidad. El inconveniente de esta técnica reside en que al aplicar el filtro solamente sobre las páginas publicitarias, se produce un suavizado en estas que hace que varíen los gradientes de la imagen y que lleguen a ser diferentes a los del logotipo. Se podría pensar que si aplicáramos también el filtro a los logotipos, se obtendrían dos representaciones suavizadas que se asemejaran, sin embargo esto no es así. En primer lugar no tiene sentido buscar dos representaciones suavizadas del logotipo y de la página publicitaria, en ese caso es mejor utilizar las representaciones originales. En segundo lugar, debido a la diferencia de tamaño entre los logotipos y las imágenes, el suavizado también difiere. No es lo mismo realizar un suavizado tomando un kernel de 5×5 sobre un logotipo de 300×300, que aplicarlo sobre una zona de 20×20 en una imagen de 2300×3000. En conclusión esta técnica tampoco introduce ninguna mejora, por lo que queda descartada para el sistema final

Por otro lado se han estudiado otras dos técnicas de pre-proceso, relacionadas con la gestión del color. Estas técnicas son *PCA* y *RGB Decomposition,* cuyo objetivo consiste en resaltar o incrementar el contraste de los logotipos utilizando la información de color de los mismos. El análisis *PCA* aplicado a esta tarea no ha resultado satisfactorio para el sistema. Se ha demostrado que el coste computacional de esta técnica es muy elevado puesto que es necesario proyectar cada uno de los colores representativos de cada uno de los logotipos, contra todas las páginas publicitarias de la base de datos. Este proceso conlleva un coste de $3 \times (2\alpha N + \beta(N \times M))$, con $N \ll M \ \wedge \ \alpha, \beta$ los coeficientes del coste de extracción de puntos ($\alpha = 0.05, \beta = 0.95$). Este coste es exponencial con la talla de N y M por lo que se hace inviable a pesar de aportar mejoras en la detección de logotipos. En este sentido, la técnica *RGB Decomposition* ofrece unos resultados similares e incluso mejores, con un coste computacional muchísimo menor. El coste de aplicar *RGB Decomposition* se reduce a $3 \times (2\alpha N + \beta M)$ que es lineal. En el capítulo 4.2.2.3 se ha presentado la comparativa de las técnicas escala de grises, *PCA* y *RGB Decomposition*, sobre el *Phantom*. Los resultados reflejan que *PCA* mejora la detección de logotipos pero a costa de un elevado precio. Se consigue incrementar la tasa de verdaderos positivos en torno a un 25%, y se mejoran notablemente los estimadores *sensibilidad, precisión* y *accuracy*. Sin embargo, el coste temporal aumenta en torno a 100 veces más con respecto al análisis en escala de gris, y sobre 35 veces más con respecto al análisis *RGB Decomposition* para nuestro caso concreto. La mejora de los resultados es sustancial, sin embargo *RGB Decomposition* iguala esta mejora con un coste muchísimo menor. Al igual que *PCA*, *RGB Decomposition* mejora en torno a un 25% la tasa de *verdaderos positivos*. En concreto, sobre el experimento realizado contra el *Phantom,* se pasa de detectar 19 *verdaderos positivos* en escala de grises, a 25 de los 41 existentes en el *Phantom*. El análisis en descomposición de color también reduce ligeramente el número de *falsos positivos,* y por lo que respecta a los estimadores calculados en el análisis, se mejoran todos y cada uno de ellos con respecto a *PCA* y al análisis en escala de grises. Para finalizar, *RGB Decomposition* únicamente incrementa en 2.5 veces el coste temporal con respecto al análisis en escala de grises. La cota superior que puede alcanzar este incremento del coste temporal es 3 veces más, puesto que en *RGB Dececomposition* se calculan

puntos interesantes en los 3 canales de color. Sin embargo, es fácil que en la descomposición de color, en alguna de las capas no se retenga mucha información y por lo tanto no se extraigan muchos puntos interesantes. Por este motivo, el coste temporal medio se sitúa en torno a $2.5 \leq 3$. Se puede observar también en la comparativa realizada sobre las diferentes técnicas de gestión de color, que *RGB Decomposition* es la técnica que consigue maximizar la función objetivo definida En definitiva, tras el análisis de los resultados arrojados por estas técnicas, queda claro que *RGB Decomposition* es la que presenta un mejor balance entre calidad de resultados y coste computacional de aplicar la técnica.

Aún con todo esto, es necesario evaluar desde el punto de vista real la utilidad o necesidad de aplicar estas técnicas que incrementan considerablemente el coste del sistema. Si observamos los resultados obtenidos tras analizar base de datos *Test* Independiente aplicando únicamente la conversión a escala de grises, observamos que se alcanza una tasa de acierto del 85%, es decir se logran detectar prácticamente todos los logotipos. Lo que se extrae de este razonamiento es que, a pesar de que el coste computacional de aplicar la técnica *RGB Decomposition* no es excesivamente caro y que la mejora en los resultados sobre el *Phantom* es sustancial, es posible que dicha técnica no sea necesaria de cara a un análisis de una base de datos real. En las bases de datos Mono-Medio y Multi-Medio, *RGB Decomposition* no conseguiría mejorar los resultados. El único logotipo que no ha sido posible detectar se corresponde con el mismo caso para ambas bases de datos, y presenta una excesiva compresión y baja calidad que hace imposible su detección. *RGB Decomposition* únicamente podría reforzar los *matchings* ya detectados con mayor cantidad de puntos. Para la base de datos *Test Independiente* sí se conseguiría mejorar los resultados. En esta base de datos, el análisis sin pre-procesos genera 7 *falsos positivos*, de los cuales 3 podrían detectarse mediante la técnica *RGB Decomposition*. En este caso, la mejora si se puede valorar, pero aun así la tasa de acierto obtenida en escala de grises se encuentra sobre el 85% que es muy elevada. En conclusión, la técnica *RGB Decomposition* se delegará a un uso opcional cuando se desee una precisión en los resultados superior. En ese sentido, la tasa media de acierto del sistema en escala de grises se situará sobre el 93%, mientras que con *RGB Decomposition* podremos incrementarla hasta en torno a un 96% de acierto, asumiendo por supuesto el incremento del coste del análisis.

Por otra parte, ha sido también necesario hacer un estudio de los parámetros óptimos de SURF para ajustar al máximo el rendimiento del sistema. Dicho estudio se ha realizado para la plataforma MATLAB® y para la librería IRIS, obteniéndose configuraciones diferentes para ambos. A priori, asumiendo que las implementaciones fueran iguales, deberíamos haber obtenido las mismas configuraciones óptimas para ambas plataformas, sin embargo sí existen diferencias de implementación, precisión y arquitectura, que hacen que varíen los resultados. En definitiva y a la vista de los resultados, la implementación en C# del algoritmo SURF presenta unas condiciones de estabilidad y regularidad mejores que MATLAB®. Si observamos las gráficas de la respuesta de la función objetivo, se puede ver claramente como la respuesta de IRIS (C#) varía de forma más suave, sin tantos saltos bruscos como los que presenta MATLAB®. Esto indica unas condiciones de estabilidad de la implementación de IRIS, que aseguran que la configuración óptima obtenida puede trabajar en un rango más amplio de situaciones que la de MATLAB®. Si observamos las configuraciones finales obtenidas, la configuración para IRIS es mucho menos exigente en cuanto a coste computacional que la obtenida por MATLAB®, para satisfacer la misma función objetivo. A mayores valores de *octaves* y menores valores de *threshold*, el coste computacional de extracción de puntos aumenta. IRIS necesita de una octava menos y un orden de magnitud más de *threshold* para obtener la misma calidad de resultados que MATLAB®. En cuanto a tiempo, obviamente C# es mucho mejor que MATLAB® por ser lenguaje pseudo-compilado. Además, el tiempo de extracción de puntos crece exponencialmente conforme se aumenta las octavas y se disminuye el *threshold*, por lo que la configuración de MATLAB® está más expuesta a este incremento de coste.

En conclusión, por cuestiones de calidad de implementación del algoritmo SURF, y arquitectura y precisión de las plataformas en la que se ejecuta, la implementación de C# es más potente que la de MATLAB®.

En cuanto a los diferentes modelos de clasificación propuestos para la eliminación de los *falsos positivos*, finalmente se ha elegido el número de puntos del *matching* y el IQR de la distribución del ratio de distancias como variables de dicho modelo. Ambas variables se han normalizado aplicado el logaritmo natural, y en cuanto al IQR, las distancias entre los puntos se han calculado mediante la distancia *city block* o $\mathcal{L}_1$ ya que conseguimos mayor separabilidad entre las clases. La validación del modelo mediante el *test independiente* nos afirma el buen diseño del mismo. Únicamente se clasifican incorrectamente 4 *verdaderos positivos*, de los cuales 1 de ellos es posible corregirlo cuando se entrene el modelo con todas las muestras. En cuanto a los 3 restantes, todos ellos son casos de muy baja calidad que, a pesar de conseguir *matchear* correctamente algunos puntos sobre el logotipo, otros caen fuera del mismo produciendo una distribución de distancias incorrecta.

Por último, con respecto a los análisis de bases de datos reales, los resultados obtenidos avalan la calidad del sistema. En cuanto a *sensibilidad*, el peor valor que arroja el sistema es un 84.44% referente al *Test Independiente*, mientras que el mejor valor alcanza el 98.11% de la base de datos *Multi-Medio*. Con respecto al *Test Independiente*, es posible corregir 3 de los *falsos negativos* mediante la aplicación de la técnica *RGB Decomposition,* en cuyo caso aumentaría la *sensibilidad* del sistema a 93.33% detectándose finalmente 42 logotipos de 45 existentes. En cuanto al ratio de *falsos positivos* y la *especificidad*, el clasificador *QDA* cumple con el objetivo para el cual se ha diseñado, eliminando en torno a un 92% de estos casos. De cara a un uso real, el trabajo llevado a cabo por este clasificador es de vital importancia. En el capítulo 4.2.5.3 se ha presentado la comparativa de aplicar el clasificador *QDA* a los resultados, o no aplicarlo. La utilidad principal de este clasificador reside en el número de resultados que deberíamos comprobar, por ejemplo de forma manual, si queremos cerciorarnos de los *verdaderos positivos* que se han detectado. Por ejemplo en IRIS, sobre la base de datos Mono-Medio, sin *QDA* generamos $26 + 96 = 122$ resultados que se deberían comprobar manualmente para confirmar la verosimilitud de los resultados. Aplicando *QDA* reducimos drásticamente esta búsqueda a $26 + 3 = 29$ resultados. El trabajo ahorrado es considerable. Si hacemos el mismo análisis sobre la plataforma MATLAB® los resultados aún son más favorables. Pasaríamos de comprobar $26 + 214 = 240$ resultados, a $26 + 8 = 34$ resultados. Sobre las otras bases de datos el comportamiento es similar. Esta reducción es de gran valor, puesto que los recursos necesarios para comprobar la calidad de los resultados son muchísimo menores si aplicamos el clasificador *QDA*. Por otro lado, el objetivo primordial de este post-proceso, es no eliminar *verdaderos positivos* previamente *matcheados*. Únicamente en el caso del *test independiente* no se cumple esta condición, sin embargo ya se ha comentado que estos casos son de muy baja calidad y difíciles de *matchear* en cualquier situación.

En conclusión, se ha presentado un sistema de reconocimiento de logotipos y marcas novedoso y robusto, para la ayuda a la evaluación del impacto publicitario que tiene una empresa en los medios de comunicación. Finalmente, se han superado todos los requisitos funcionales establecidos en el proyecto, proporcionando una herramienta fiable, versátil y potente que se encuentra ya actualmente dando servicio a las necesidades de los clientes.

6 BIBLIOGRAFÍA

[1] M. S. Lew, N. Sebe, C. Djeraba y R. Jain, «Content-based Multimedia Information Retrieval: State of the Art and Challenges,» *ACM Transactions on Multimedia Computing, Communications and Applications,* pp. 1-26, 2006.

[2] R. Datta, D. Joshi, J. Li y J. Z. Wang, «Image Retrieval: Ideas, Influences, and Trends of the new age,» *ACM Computing Surveys,* pp. Vol 40, No. 2, Article 5, Abril 2008.

[3] R. Z. Z. L. L. S. Xingxing Chen, «A Multi-scale Phase Method for Content Based Image Retrieval,» *Fourth International Conference on Image and Graphics ICIG,* pp. 795-800, 2007.

[4] C. Harris y M. Stephens, «A combined corner and edge detector,» *In Proceedings of the Alvey Vision Conference,* pp. 147-151, 1988.

[5] T. Lindeberg, «Feature detection with automatic scale selection.,» *IJCV 30(2):,* pp. 79-116, 1998.

[6] K. Mikolajczyk y C. Schmid, «Indexing based on scale invariant interest points.,» *In ICCV,* vol. 1, pp. 525-531, 2001.

[7] D. Lowe, «Object recognition from local scale-invariant features.,» *In ICCV,* 1999.

[8] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding (CVIU),* pp. Vol. 110, No. 3, pp 346-359, 2008.

[9] K. Mikolajczyk y C. Schmid, «A Performance Evaluation of Local Descriptors,» *IEE Transactions of pattern analysis and machine intelligence,* pp. Vol. 27, No. 10, 2005.

[10] A. D. Bagdanov, L. Ballan, M. Bertini y A. Del Bimbo, «Trademark Matching and Retrieval in Sports Video Databases,» *Proceedings of the international workshop on Workshop on multimedia information retrieval - MIR '07,* pp. 79-86, 2007.

[11] L. Ballan, M. Bertini, A. Del Bimbo y A. Jain, «Automatic trademark detection and recognition in sport videos,» *ICME,* pp. 901-905, 2008.

[12] T. Fawcett, «An introduction to ROC Analysis,» *Pattern Recognition Letters 27,* p. 861–874, 2006.

[13] J. F. Canny, «A Computational Approach To Edge Detection,» *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, nº 6, p. 679–698, 1986.

[14] R. Maini y H. Aggarwal, «Study and Comparison of Various Image Edge Detection Techniques,» *International Journal of Image Processing (IJIP),,* vol. 3, nº 1, pp. 1-12.

[15] N. Wiener, Extrapolation, Interpolation, and Smoothing of Stationary Time Series, New York: Wiley. ISBN 0-262-73005-7., 1949.

[16] A. Kolmogorov, «Stationary sequences in Hilbert space,» vol. 2, nº 6, pp. 1-40, 1941.

[17] K. Pearson, «On Lines and Planes of Closest Fit to Systems of Points in Space,» *Philosophical Magazine,* vol. 2, nº 6, p. 559–572, 1901.

[18] D. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints,» *International Journal of Computer Vision,* pp. 1-28, 2004.

[19] C. Evans, «Notes on the OpenSURF library,» pp. 1-25, 2009.

[20] P. Viola y M. Jones, «Rapid Object Detection using a Boosted Cascade of Simple Features,» *Conference on computer vision and pattern recognition,* pp. 1-9, 2001.

[21] H. Bay, T. Tuytelaars y L. Van Gool, «Surf: Speeded up robust features,» *Computer Vision–ECCV,* pp. 404-417, 2006.