ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

INGENIERÍA DE TELECOMUNICACIÓN - INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

PROYECTO FIN DE CARRERA

Algoritmos de detección y descripción de características visuales para el reconocimiento de objetos. Análisis y aplicaciones.

Autor: Bernard Hernández Pérez
Tutor: Inmaculada Mora Jiménez

Curso académico: 2012/2013

Proyecto Fin de Carrera

Algoritmos de detección y descripción de características visuales para el reconocimiento de objetos. Análisis y aplicaciones.

Autor: Bernard Hernández Pérez
Tutor: Inmaculada Mora Jiménez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2013, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2013

"The beautiful thing about learning is nobody can take it away from you."B. B. King

Agradecimientos

Esta es la primera vez que hago un trabajo tan extendido y que representa el fin-comienzo de una nueva etapa de mi vida, por lo que quiero expresar mi gratitud a todos quienes, de una u otra manera, me han acompañado en esta larga jornada.

Agradecer a la Universidad Rey Juan Carlos, por haberme ofrecido la oportunidad de estudiar esta doble titulación que facilitará mi futura carrera profesional. Además, por haber hecho posible las dos estancias de intercambio realizadas, en Suecia y Francia, que han conseguido enriquecerme como persona. A todos los profesores, que con su dedicación y empeño, intentaron y, quiero creer, consiguieron, transmitirme parte de sus conocimientos. Y a todos los amigos, con los que tantos buenos momentos y alegrías he compartido. Espero que todos ellos sepan que han sido, y son, una parte importante de mi vida.

Quiero agradecer a mi padre, Isidro Hernández Rodriguez, a mi madre, Angelina Pérez Muñoz y a mi hermano, Mario, mi familia, quienes participaron, directa e indirectamente, de mi formación. Especialmente a mis padres, que pudieron y quisieron darme el privilegio de estudiar. Sin vosotros esto no habría sido posible.

Quiero expresar mi agradecimiento a mis abuelos, Bienvenido Hernández y Ana María Rodriguez, a quienes siempre preferí llamar yayos, y a mi abuela Angelina Muñoz, por ser unos abuelos excepcionales, que ayudaron en mi crianza, aceptaron y fomentaron mis gustos y compartieron conmigo sus experiencias.

Finalmente a Alicia, por su amor, comprensión, paciencia y fortaleza. Con quien tantas experiencias he compartido y con total seguridad compartiré. A la que quiero recordar, como me pidió en París, lo felices que somos cuando estamos juntos. *"El hombre que siente mucho habla poco, y en un beso, sabrás todo lo que ha callado."*

Cada uno de ustedes, directa e indirectamente, ha sido fundamental en la realización de esta tesis, por lo tanto, responsables de ella. Sólo les libro de los errores y omisiones de la memoria que mi escritura pudiese conllevar.

Resumen

La *visión artificial* es la parte de la inteligencia artificial que incluye métodos para la adquisición, procesamiento, análisis y comprensión de imágenes para obtener resultados de forma númerica o simbólica. La información proporcionada por los resultados es usada para tomar decisiones. No podemos hablar de visión artificial de forma aislada, las relaciones e interacciones con otros campos es inevitable y merecen especial mención el *procesamiento de imágenes*, *reconomiento de patrones* y el *aprendizaje máquina*.

El principal objetivo de este proyecto es analizar el comportamiento de los algoritmos de extracción de *características visuales* existentes y su eficacia en la toma de decisiones. La detección de un objeto en la imagen, su clasificación y reconocimiento son el tipo de decisiones que estudiaremos.

En el proyecto se aplicarán los algoritmos anteriores al reconocimiento de objetos con una vista principal, a la que denominaremos *vista canónica*. Se analizará la *robustez* de estos algoritmos frente a determinadas *transformaciones* del objeto capturado en distintas imágenes, así como el *tiempo de cómputo* empleado para la *detección*, *descripción* y *emparejamiento* de dichas características visuales y sus prestaciones en el reconocimiento de imágenes en bases de datos.

Debido al auge de los *dispositivos móviles* y la facilidad de su uso, propondremos algunas aplicaciones y analizaremos la viabilidad de estos algoritmos para el desarrollo de *aplicaciones móviles*.

Índice General

Índice de Figuras

Índice de Tablas

Acrónimos

API	**A**pplication **P**rogramming **I**nterface
ASIFT	**A**ffine **S**cale **I**nvariant **F**eature **T**ransform
BGR	**B**lue **G**reen **R**ed
BRIEF	**B**inary **R**obust **I**ndependent **E**lementary **F**eatures
BSD	**B**erkeley **S**oftware **D**istribution
CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
CV	**C**omputer **V**ision
DoB	**D**ifference **o**f **B**oxes
DoG	**D**ifference **o**f **G**aussians
DSF	**D**jango **S**oftware **F**oundation
FLANN	**F**ast **L**earning **A**pproximated **N**earest **N**eighbors
GMM	**G**aussian **M**ixture **M**odel
GPL	GNU **G**eneral **P**ublic **L**icense
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphics **P**rocessing **U**nit
HTML	**H**yper**T**ext **M**arkup **L**anguage
KNN	**K** **N**earest **N**eighbors
LoG	**L**aplacian **o**f **G**aussian
MoG	**M**ixture **o**f **G**aussians
MSER	**M**aximally **S**table **E**xtremal **R**egions
ORB	**OR**iented **B**RIEF
PSF	**P**ython **S**oftware **F**oundation

Capítulo 1

Introducción

En la presente memoria se analizan algunos algoritmos de detección y descripción de características visuales en imágenes. Estas características son usadas en aplicaciones relacionadas con *visión artificial* [1] para la detección, clasificación y/o reconocimiento de los objetos que aparecen en dichas imágenes.

La memoria está estructurada de la siguiente manera. En el Capítulo 1 acercaremos al lector al contexto general del proyecto fin de carrera, la *visión artificial*, ilustraremos su utilidad con aplicaciones y comentaremos brevemente las fases del proyecto. El capítulo se completa presentando las herramientas utilizadas para su desarrollo. En el Capítulo 2, *Motivación y objetivos*, se comentarán las motivaciones que originaron este trabajo, se detallará el propósito de forma más precisa, las limitaciones a afrontar y los entregables. En el Capítulo 3, *Materiales y métodos*, nos adentraremos con más detalle en la *visión artificial* y explicaremos tanto conceptos básicos de tratamiento de imágenes como trabajos realizados recientemente por otros investigadores en este área, lo que se conoce como *estado-del-arte*[2]. En el Capítulo 4, *Resultados y análisis*, presentaremos algunos resultados con experimentos controlados, su análisis y las decisiones tomadas en base a los mismos. Seguiremos presentando el desarrollo de aplicaciones particulares en el Capítulo 5, donde se especificará el esquema de implementación de cada una de las aplicaciones parciales y las decisiones de diseño. Finalmente, en el Capítulo 6, *Conclusiones*, presentaremos las conclusiones y las posibles líneas a seguir para futuros trabajos.

El presente capítulo comienza con las aplicaciones de *visión artificial* existentes y que, en gran medida, nos llevaron a la elección de la *visión artificial* como

[1] Subcampo de la inteligencia artificial con el propósito de programar un computador para que entienda una escena o las características de una imagen.

[2] Anglicismo derivado de la expresión *State of the art*, muy utilizado en investigaciones y consultas. Se refiere a los aspectos más avanzados relacionados con la tecnología.

tema principal de este proyecto fin de carrera. Seguiremos con el planteamiento genérico del problema a abordar y las actividades que han de ser llevadas a cabo. Concluiremos este capítulo describiendo brevemente las herramientas empleadas para el desarrollo de este proyecto.

1.1. Contexto general

La *visión artificial* o visión por computador (CV) persigue reproducir algunas de las capacidades del sistema visual humano, con el fin de proporcionar a las máquinas la capacidad de ver, reconocer e interpretar imágenes [1] [2]. Para los humanos la vista es, sin duda alguna, uno de los sentidos más importantes y está siendo cada vez más explotado, dentro de sus limitaciones, por diversos sistemas informáticos.

Para enfatizar la importancia de la visión por computador, no sólo en el área académica sino también en la vida cotidiana, presentamos a continuación algunas aplicaciones [3] que hacen uso de los descubrimientos y avances logrados por diversos investigadores en dicho campo.

A) Videojuegos

Este es un sector clave y que ha contribuido notablemente al desarrollo de la visión por ordenador. Ejemplos claros de su uso son el dispositivo *Kinect*, desarrollado por *Microsoft* ®, y *Eye Toy*, desarrollado por *Play Station* ®, para el reconocimiento de los gestos realizados por los jugadores. En la nueva generación de consolas de videojuegos se está avanzando y poniendo mucho esfuerzo en facilitar la interacción jugador-consola con dichos dispositivos. El análisis de la mirada, *eye tracking*, se incorpora para estimar los lugares de la pantalla a los que los jugadores están prestando más atención para, por ejemplo, ayudarles a apuntar a su objetivo en los juegos de disparos [4] [5]. Una empresa relativamente nueva que se dedica a dicho análisis es *Tobii* ® [3].

B) Robótica

En el proceso de desarrollo de un robot, una de las partes más complicadas es interpretar el mundo a su alrededor a través de la información proveniente de su entorno [6]. Esta información puede ser capturada por diversos instrumentos, como sensores o cámaras de vídeo, y puede usarse para la localización/reconocimiento de objetos y/o el seguimiento de los mismos.

C) Compra visual

Este campo está en desarrollo y consiste en tomar una foto de un producto en una tienda, para buscar en internet información sobre dicho producto o

[3] Link Tobii: http://www.tobii.com/

similares y así comparar sus características, calidad y precio. Una empresa innovadora en este sector es *OculusAI* ® [4] con su aplicación móvil Productify ®.

D) Medicina

El objetivo básico de la visión por ordenador en este área es el tratamiento y análisis de imágenes, detección de patrones y en ocasiones reconstrucción 3D, para facilitar y ayudar al correcto diagnóstico por parte del especialista clínico. Aplicaciones comunes son la detección y caracterización automática de tumores, análisis de mamografías digitales, mejora de la imagen de microscopía o análisis de imagen hiperespectral para extraer la composición de los elementos contenidos en una imagen. También se usan de forma más compleja en cirugía como herramienta de ayuda, por ejemplo, estimando el movimiento de la superficie del cerebro en intervenciones quirúrgicas.

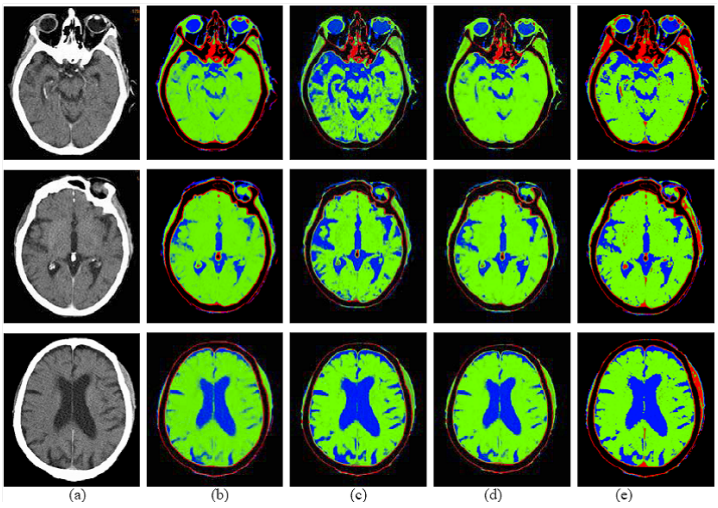


Figura 1.1: Segmentación del cerebro en imágenes CT. Imágenes capturadas con un escaner CT, que genera una imagen bidimensional de una sección perteneciente a un objeto tridimensional. Rojo para huesos de baja intensidad, verde para materia cerebral y azul para fluidos. Podemos observar a la izquierda la imagen original y el resultado de aplicar diferentes métodos de segmentación en la imagen. Imagen tomada de *Soft segmentation of CT brain data* [7].

E) Sistemas de seguridad

En este ámbito existen numerosas aplicaciones. Podemos hablar de sistemas de control de acceso a áreas restringidas, detectando intrusos de manera automática mediante análisis de las imágenes grabadas por una cámara de vídeo

[4]Link OculusAI: http://oculusai.com/

sin necesidad de que un vigilante esté observando la grabación. Además, hay sistemas que no sólo detectan intrusos sino que además pueden identificarlo comparando su rostro con los de una base de datos. Existen sistemas que detectan objetos abandonados o extraños, generalmente en el interior de edificios [8].

No sólo hemos de pensar en sistemas de seguridad a gran escala, también podemos restringir el acceso a extraños, por ejemplo, a nuestro ordenador, mediante un sistema de reconocimiento de rostros o huellas dactilares.

F) Lectura automática y reconocimiento de objetos

Su uso es muy extendido en controles de aduanas y aeropuertos, para detectar y reconocer objetos que puedan ser peligrosos o estén prohibidos por las leyes vigentes. También sirve para recopilar información y estadísticas, desde el número de usuarios de metro, analizando las imagenes grabadas, hasta la información personal de los usuarios escaneando su DNI o pasaporte. Uno de los usos más extendidos son los lectores de códigos de barras de los productos en supermercados y códigos QR [5]. También cabe destacar su uso para los billetes electrónicos de avión en el aeropuerto.

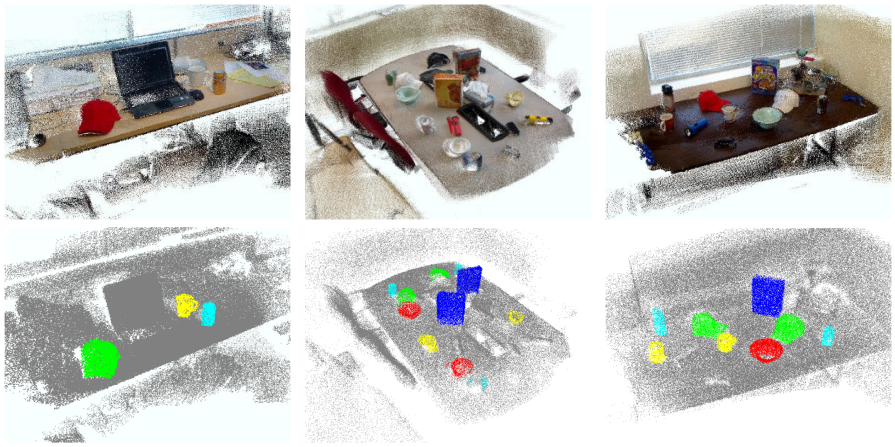


Figura 1.2: Resultado del etiquetado de elementos para tres escenas 3D complejas. Arriba se observa la reconstrucción 3D y abajo el etiquetado de los objetos. Los objetos coloreados según sus categorías son tazón o cuenco en rojo, gorra en verde, caja de cereales en azul, jarra de café en amarillo y lata de soda en cyan. Imagen tomada de *Detection-based object labeling in 3D scenes* [9].

Los anteriores son sólo algunos ejemplos de aplicaciones y empresas que se dedican a la investigación y desarrollo de sistemas de *visión artificial*. Para más

[5] QR code es el nombre de una marca comercial de un tipo de código de barras bidimensional.

información, consúltese el listado[6] de empresas que desarrollan productos de visión por computador categorizadas por su principal área de aplicación.

1.2. Fases del proyecto

En este proyecto fin de carrera se estudian, comparan e implementan varios métodos de reconocimiento de objetos presentes en imágenes. El objetivo final es desarrollar una aplicación que haga uso de estos métodos. Puesto que el abanico de posibilidades es muy amplio, se detallará un listado que contendrá las aplicaciones a realizar y posibles ideas para aplicaciones futuras. En la medida de lo posible los métodos han de ser evaluados y adaptados para aplicaciones en tiempo real.

Las siguientes tareas han sido llevadas a cabo durante la realización de este proyecto:

1. Estudio de la literatura relacionada con los métodos de detección y descripción de características visuales y los conceptos vinculados con la *visión artificial*, destacando segmentación y sustracción de fondo.

2. Comparación y evaluación de diferentes tipos de detectores y descriptores de características visuales. Selección de las características más adecuadas para abordar el problema de reconocimiento de objetos.

3. Creación de bases de datos propias con las que hacer un análisis controlado, así como uso de bases de datos ya existentes y de mayor tamaño que las propias. Se hará un esfuerzo por establecer conclusiones sobre las prestaciones de los algoritmos de detección y descripción en estas bases de datos.

4. Desarrollo de una aplicación. Tendremos como entrada una imagen con el objeto a reconocer y haremos uso de una base de datos con los objetos previamente etiquetados. Elegiremos el objeto de la base de datos que tenga mayor número de concordancias con la imagen de entrada.

5. Redacción de la memoria.

1.3. Herramientas

El trabajo relacionado con este proyecto fue realizado en un ordenador portatil MacBook Pro con un procesador Intel i7@2.9GHz y 8 GB RAM DDR3. Es difícil estimar la cantidad de horas dedicadas al proyecto, lo indicamos de forma aproximada: la mitad del tiempo fue destinado a la lectura y comprensión de los conceptos teóricos (incluyendo la revisión del estado del arte), un cuarto al

[6] Link listado: http://www.cs.ubc.ca/~lowe/vision.html

desarrollo de los experimentos y la implementación de las aplicaciones y el cuarto restante a redactar la memoria. Las herramientas utilizadas para el desarrollo del proyecto final de carrera son descritas brevemente a continuación. Nótese que, en la medida de lo posible, se han usado tecnologías y proyectos de código abierto y de uso libre. Nuestro objetivo no es describir las herramientas en detalle. A tal fín, enlaces a información adicional son dados cuando se considera necesario.

A) Python

Python es un lenguaje de alto nivel que enfatiza la facilidad de lectura del código. Es un lenguaje dinámico, es decir, los tipos de datos se asignan en tiempo de ejecución y es generalmente usado como un lenguaje para realizar *scripts*. Soporta los paradigmas de programación orientada a objetos, imperativa y funcional. Python es libre, de código abierto y está mantenido por la PSF (*Python Software Foundation*). Gran parte de las figuras del Capítulo 4, *Resultados y análisis*, fueron obtenidas usando Python [10] y la librería de creación de gráficos *Matplotlib*.

B) Matlab

MATLAB[7] (***mat****rix* ***lab****oratory*) es un entorno para realizar operaciones numéricas desarrollado por *MathWorks*. Permite la manipulación de matrices, representación de funciones y datos, creación de interfaces de usuario e interacción con programas escritos en otros lenguajes. Tiene licencia propietaria y actualmente está disponible la versión 2013a para Windows, Linux y Mac OS.

C) OpenCV

OpenCV[8] (*Open Source Computer Vision*) [11] es una librería de visión artificial desarrollada inicialmente por *Intel* ®, que actualmente es de código abierto y está publicada bajo licencia BSD, lo que permite su libre utilización en propósitos comerciales o de investigación. Esta librería proporciona un extenso conjunto de funciones para trabajar con *visión artificial*, cuyo desarrollo se ha realizado primando la eficiencia para uso en aplicaciones en tiempo real. Por ese motivo, está desarrollada en los lenguajes C/C++, optimizados y haciendo uso de varios núcleos. Puede hacer uso de primitivas de rendimiento integrado de los procesadores *Intel* ®. Tiene interfaces para C, C++, Python y Java y soporta los siguientes sistemas operativos: Windows, Linux, Mac OS, iOS y Android.

D) OpenLibrary

OpenLibrary[9] es un proyecto sin ánimo de lucro cuyo código, documentación y datos son abiertos. Fue creada por Aaron Swartz. El objetivo de esta base

[7] Link MATLAB: http://www.mathworks.com/
[8] Link OpenCV: http://opencv.org/
[9] Link OpenLibrary: http://openlibrary.org/

de datos es almacenar información de cada libro publicado. Esta biblioteca digital ha sido fundada gracias a una beca de la *California State Library* y la *Kahle/Austin Foundation*. Aceptan contribuciones de cualquier usuario y disponen de una RESTful API para acceder, de forma sencilla, a toda la información almacenada.

E) JustInMind

JustInMind ® [10] es una herramienta de desarrollo de prototipos para aplicaciones móviles y *wireframes* para webs. Ofrece la posibilidad de importar/exportar *widgets*, definir interacciones y simular la aplicación. Además, soporta simulaciones de alta fidelidad, generando el código HTML con contenido *JavaScript* o *Adobe Flash*. Actualmente está disponible la versión 5.1 para Windows y Mac OS. Fue premiado con el *Best Application Award* en la *EclipseCon 2012*.

F) Django

Django[11] es un framework para desarrollo web libre y de código abierto. Está escrito en Python y sigue el modelo de control de vistas. Facilita el uso de las bases de datos y agiliza el proceso de desarrollo web. Además, ofrece de forma opcional un interfaz de administración sencillo y fácil de usar por los *webmasters*. Es mantenido por la DSF (*Django Software Foundation*).

G) Shogun

The Shogun Machine Learning Toolbox[12] es una librería que ofrece numerosos algoritmos y estructuras de datos para problemas de *Machine Learning*, es decir, aprendizaje máquina. Está desarrollada en C++, publicada bajo licencia GNU y su creador es Soeren Sonnenburg [12]. La página web de Shogun fue una contribución realizada por mí durante el verano de 2012.

[10] Link JustInMind: http://www.justinmind.com/
[11] Link Django: http://www.djangoproject.com/
[12] Link Shogun: http://shogun-toolbox.org/

Capítulo 2

Motivación y objetivos

En este proyecto tendremos una imagen de entrada a nuestro sistema de reconocimiento, imagen que contendrá uno o más objetos. Para reconocer si esos objetos están presentes en nuestra base de datos, extraeremos sus características visuales y las compararemos con las de los objetos que previamente hemos etiquetado y almacenado.

Las bases de datos generalmente disponen de varias vistas del mismo objeto, siendo algunas de ellas más representativas que otras. Las vistas más representativas son denominadas vistas *canónicas*. El objetivo es desarrollar un algoritmo de reconocimiento robusto a cambios en la vista canónica del objeto y extrapolable a varias vistas (*multi-view*). Utilizaremos una base de datos con una vista canónica para cada objeto, del cual extraeremos características robustas a cambios de iluminación, escalado y rotación. Por último compararemos las características extraídas de una imagen con las almacenadas en una base de datos y determinaremos si el objeto se encuentra en la imagen.

Explicaremos en la Sección 2.1 la motivación, ideas y decisiones tomadas tras realizar el proceso de *brainstorming*[1]. En la Sección 2.2 detallaremos el objetivo principal del proyecto fin de carrera. Continuaremos con una explicación de los objetivos parciales en la Sección 2.3 y de las limitaciones a afrontar en la Sección 2.4. Los elementos de los que se compone el proyecto fin de carrera, que serán entregados tras su finalización, se listan en la Sección 2.5.

[1]Es una técnica de creatividad individual o grupal. En esta técnica se hacen esfuerzos para hallar soluciones a un problema específico mediante la creación de una lista de ideas propuestas de forma espontánea por los miembros.

2.1. Motivación

Hoy en día todos los dispositivos electrónicos, teléfonos móviles o tabletas, tienen una cámara integrada. Esta característica puede ser explotada por los usuarios de muchas maneras distintas. Una de ellas es la denominada *Visual shopping* o compra visual, que a partir de concordancias entre la foto de un producto y las imágenes almacenadas en una base de datos, sugiere al usuario productos similares. La idea nació a raíz de una posible colaboración con una empresa relacionada con el sector de la visión artificial, que al final no pudo ser llevada a cabo y derivó en el trabajo presentado en este proyecto.

Es deseable reconocer el producto de forma correcta sin importar la posición desde la que ha sido tomada la imagen. Se dice que *una imagen vale más que mil palabras*, pero no todas las imágenes son igual de importantes o útiles. Para una imagen concreta, las características capturadas y su relevancia varían entre diferentes vistas, y estas variaciones originan fallos en los algoritmos de reconocimiento de productos. Mejorar la estabilidad frente a posibles variaciones de la imagen, como cambio de vista, rotación o iluminación, mejorará cuantitativa y cualitativamente el reconocimiento de productos, repercutiendo de forma significativa en la experiencia del usuario. Puesto que se trata de un problema muy amplio y complicado, en este proyecto se seguirá una aproximación gradual a la solución.

El problema de reconocimiento de objetos en imágenes puede ser extendido a otras áreas como recogida de datos para realizar estadísticas, robótica, videojuegos o aplicaciones móviles entre otras. Para definir la aplicación hemos de decidir el tipo de objetos a reconocer y el entorno en el que los encontraremos. En primer lugar realizamos un proceso de *brainstorming* a partir del cual surgieron las siguientes aplicaciones:

A) Reconocimiento de logotipos

   En documentos publicitarios, y concretamente folletos y carteles, suele aparecer una sección con los logotipos de las empresas colaboradoras. En muchas ocasiones estos logotipos son desconocidos por el usuario. Una aplicación de reconocimiento de logotipos permitiría, mediante un clic, disponer de forma sencilla de la información asociada a dichas empresas. Esta información se puede combinar con otras aplicaciones de realidad aumentada, localización y GPS para mostrar dónde están situadas las empresas o una ruta para llegar a sus instalaciones. Asumiremos que el entorno o fondo es un documento de color constante en el que están impresos los objetos a reconocer, es decir, los logotipos.

   Un ejemplo del interfaz desarrollado para esta aplicación se muestra en la Figura 2.1. El cartel publicitario con los logotipos de las empresas colaboradoras se muestra en la Figura 2.1a. Tras tomar una foto de la zona inferior del

cartel, la aplicación nos mostraría un resultado similar al de la Figura 2.1b. Para desarrollar este prototipo se ha usado *JustInMind* ®[2], software para definir y crear prototipos de aplicaciones móviles y webs.

(a) Cartel (b) Logotipos

Figura 2.1: Ejemplo de aplicación de reconocimiento de logotipos. Imagen del cartel (izquierda), y resultado tras hacer la foto a la sección inferior con los patrocinadores (derecha). La herramienta de desarrollo utilizada es *JustInMind* ®.

B) Reconocimiento de cuadros de museos

La principal idea es permitir a los usuarios crear su propia guía de un museo, para que al tomar la foto de un cuadro se muestre toda la información que el usuario considera relevante. Para definir la información relevante el usuario podría indicar enlaces a páginas web de interés o escribir él mismo su propio contenido. De este modo esta aplicación podría ser muy útil para crear material educativo de forma dinámica y compartida. Asumiremos que el entorno o fondo es la pared, de color constante, en la que se encuentran los objetos a reconocer, es decir, los cuadros. Un posible interfaz para la aplicación, desarrollada con *JustInMind*, se muestra en la Figura 2.2.

C) Reconocimiento de portadas de libros

Otra posibilidad es desarrollar una aplicación que permita tomar una foto de la portada de un libro y mostrar información complementaria. Esta informa-

[2]Link JustInMind: http://www.justinmind.com/

ción puede ser básica (autor, título, fecha de publicación) o información más relevante como libros relacionados y opiniones de la prensa especializada o de otros lectores. Toda esta información ya ha sido recopilada y almacenada en bases de datos por otros proyectos como OpenLibrary y puede ser accedida de forma sencilla mediante su RESTful API. Los objetos a reconocer serían libros y el entorno sería cualquier escenario de la vida real.

Figura 2.2: Ejemplo de aplicación de reconocimiento de cuadros tras realizar una foto a la imagen de *La Gioconda*. La herramienta de desarrollo utilizada es *JustInMind* ®.

Debido al amplio abanico de escenarios susceptibles de aplicar técnicas de reconocimiento de objetos, sería útil realizar una aplicación que permita al usuario crear de forma sencilla su propia lista con las imágenes de los objetos a reconocer y definir el contenido que se desea mostrar una vez reconocido el objeto. De este modo conseguiremos crear una aplicación versátil que puede adaptarse a las necesidades de cada usuario. Con el objetivo de permitir el uso de esta aplicación por parte de diferentes tipos de dispositivos (móviles, tablets u ordenadores) en diferentes sistemas operativos (iOS, Android, Windows o Symbian entre otros) se ha decidido que la mejor opción es realizar una aplicación web usando Django. El paradigma de diseño escogido se conoce como *Cloud Computing*, computación en la nube, y permite ofrecer servicios de computación a través de internet. Todos los conceptos y decisiones se presentarán y justificarán posteriormente.

2.2. Objetivo principal

El objetivo principal de este proyecto fin de carrera es proponer, diseñar y desarrollar un sistema para el reconocimiento visual de objetos en tiempo real. Entre las aplicaciones consideradas en la primera sección, hemos decidido centrarnos en el reconocimiento de libros por las siguientes razones:

- Los libros disponen de una sóla vista canónica y plana, a la que denominaremos portada, que simplifica el proceso de reconocimiento.
- Las portadas contienen una gran cantidad de información y detalles, que facilitarán su identificación.
- Es sencillo crear nuestra propia base de datos con portadas y/o localizar bases de datos previamente creadas.

Para evaluar el problema de reconocimiento en distintos escenarios realizaremos dos aplicaciones, un basada en imágenes y otra basada en vídeo. Las dos aplicaciones son explicadas brevemente a continuación:

A) Reconocimiento de portadas de libros en imágenes

   La imagen de entrada al sistema de reconocimiento debe tener, al menos, una portada. Como primera fase, extraeremos de la imagen cada una de las portadas y compararemos de forma individual cada una de ellas con las portadas almacenadas en una base de datos. Para cada una de ellas indicaremos el libro al que hace referencia.

B) Reconocimiento de portadas de libros en vídeo

   En este caso dispondremos de un flujo continuo de imágenes donde puede aparecer o no la portada de un libro. De este modo podremos evaluar si es viable la realización de una aplicación en tiempo real. Combinando varias imágenes modelaremos el fondo para poder detectar el libro cuando lo situemos frente a la cámara. A continuación compararemos la portada detectada con las portadas almacenadas en una base de datos e indicaremos el libro al que hace referencia.

2.3. Objetivos parciales

Para conseguir el objetivo principal, tendremos que realizar una serie de pruebas y aplicaciones parciales que permitan verificar el correcto funcionamiento del software y la viabilidad del desarrollo de las distintas aplicaciones mencionadas en la sección anterior. Estos objetivos parciales los separamos en cuatro grupos bien diferenciados, que explicaremos a continuación.

A) Extracción de características

En este proceso seleccionaremos los puntos de la imagen que son interesantes para su reconocimiento, denominados *puntos de interés*. Para cada uno de los puntos definiremos una región de interés y describiremos sus características/propiedades mediante un vector de magnitudes y orientaciones denominado *vector descriptor*. El resultado final de aplicar el proceso de extracción de características en una imagen será un conjunto de vectores descriptores, cada uno de ellos describiendo un punto de interés.

Tras la obtención del conjunto de vectores descriptores, denominado *descriptor*, se llevarán a cabo pruebas de comparación entre las diferentes técnicas de extracción disponibles, su tiempo de cómputo, robustez a transformaciones de la imagen y prestaciones. De esta manera analizaremos si tiene sentido hablar de aplicación en tiempo real y elegiremos las técnicas que mejor se adapten para cada aplicación.

B) Detección de objetos

Analizaremos si el algoritmo propuesto es viable para la detección de objetos y de qué características (tamaño de la imagen o puntos de interés) depende su correcto funcionamiento. Se dispondrá de una imagen que represente un escenario complejo con varios objetos, denominada escena, y la imagen del objeto a detectar. Al aplicar el procedimiento de extracción de características en cada imagen obtendremos dos descriptores, uno correspondiente a la imagen de la escena y otro al objeto, respectívamente. En el proceso de detección del objeto no habrá un procesado previo de la escena, y el resultado de la detección se obtendrá comparando directamente los dos descriptores, como se muestra en la Figura 2.3.

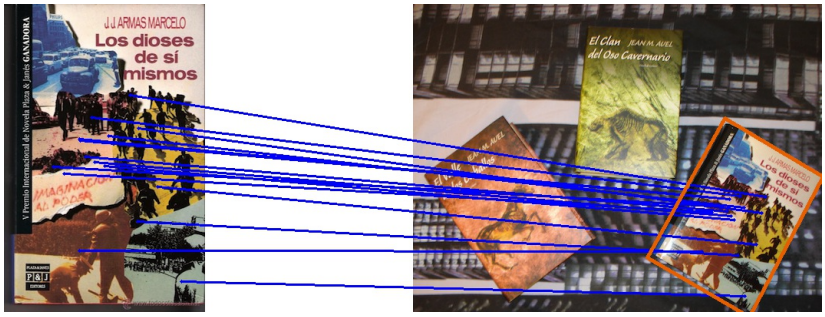


Figura 2.3: Ejemplo de detección de portadas de libros. Objeto a detectar (izquierda) y escena con el objeto (derecha).

C) Clasificación de objetos

Analizaremos si el descriptor obtenido es viable para clasificación de objetos y qué limitaciones (número de clases, variabilidad inter-clase e intra-clase) te-

nemos a la hora de realizar dicha clasificación. En el proceso de clasificación se compara el descriptor de la imagen de un objeto con los descriptores de otros objetos almacenados en una base de datos. Para cada una de esas comparaciones calcularemos el número de puntos de interés emparejados correctamente, esperando como resultado muchas correspondencias para objetos con apariencia similar. Este concepto se muestra en la Figura 2.4 con las portadas de los libros de la saga *Harry Potter* y *Shades of Gray*.

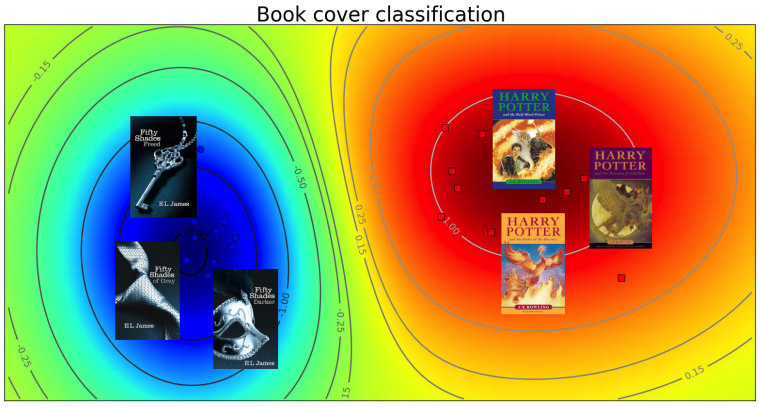


Figura 2.4: Ejemplo de clasificación de portadas de libros.

D) Reconocimiento de objetos

Evaluaremos los resultados del algoritmo de reconocimiento de imágenes con una sola vista canónica, y compararemos sus prestaciones utilizando distintos tipos de objetos. Posteriormente estudiaremos formas de mejorar el reconocimiento de objetos con varias vistas canónicas. En el proceso de reconocimiento de objetos realizaremos un procesado previo para extraer cada objeto de la escena. Posteriormente compararemos indivualmente cada objeto de la escena con el objeto a reconocer. Este matiz, mostrado en la Figura 2.5, es importante para mejorar el proceso de reconocimiento en escenas con gran número de objetos.

Figura 2.5: Ejemplo de reconocimiento de portadas de libros.

2.4. Limitaciones

La visión por computador es un concepto relativamente nuevo en el que aún no se ha llegado a resultados contundentes. Muchos investigadores están estudiando y realizando mejoras sobre algoritmos ya existentes, analizando sus prestaciones y particularizándolos según la tarea a realizar.

No hay un procedimiento claramente definido y único para abordar el problema de reconocimiento de objetos en imágenes, sino que hay varios enfoques que son igualmente válidos. La dependencia entre etapas consecutivas es muy alta, y pequeñas modificaciones en unas etapas pueden causar grandes cambios en las prestaciones del procedimiento completo.

Muchas de las implementaciones realizadas en el campo de visión por computador están patentadas, prohibiendo su uso para fines lucrativos y en ocasiones académicos o de investigación. Por ese motivo se ha decidido usar, siempre y cuando ha sido posible, implementaciones de código y uso abierto, abordando problemas de compatibilidad entre los mismos, como la definición de sus tipos básicos o el uso de distintos lenguajes de programación.

2.5. Entregables

Los elementos a entregar al terminar la realización del proyecto final de carrera son los siguientes:

- Una memoria escrita que documente el contexto del proyecto, el problema a abordar y la solución propuesta, explicando el procedimiento seguido, las pruebas realizadas y los resultados obtenidos.
- Aplicaciones o demos para cada uno de los objetivos parciales presentados anteriormente.
- Aplicación completa de reconocimiento de objetos realizada en Python y C++. Esta aplicación funcionará en local en el propio ordenador y se intentará extender usando el paradigma de computación en la nube, para facilitar el uso de la misma en dispositivos móviles.

Capítulo 3

Materiales y métodos

En la Sección 3.1 de este capítulo se presentan al lector los conceptos generales del tratamiento digital de imágenes, así como las etapas a seguir para el reconocimiento de objetos en imágenes. La Sección 3.2, *Estado del arte*, comienza con una breve intuición de los elementos básicos para el reconocimiento de objetos. A continuación explicamos las técnicas seleccionadas en este proyecto para su uso en un esquema de reconocimiento de objetos, así como su implementación, prestaciones, diferencias y limitaciones. Para un análisis exhaustivo de cada método pueden consultarse las referencias mencionadas en sus respectivas secciones.

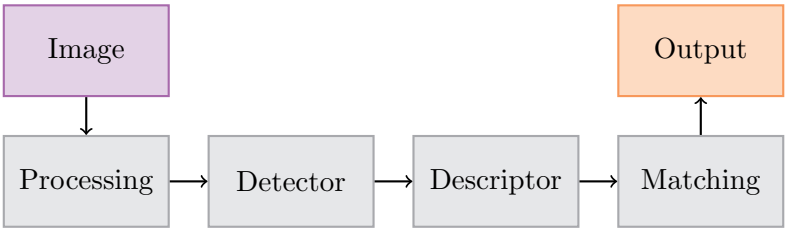


Figura 3.1: Diagrama general para el reconocimiento de objetos en imágenes. El bloque de procesado de imagen es opcional y puede mejorar las prestaciones del sistema.

Es importante señalar que no tiene sentido enumerar todas las posibles maneras de abordar este problema: cualquier combinación de los métodos explicados o modificación de los mismos conduce a un enfoque válido siempre que los resultados sean satisfactorios para nuestro propósito. Divideremos el método a seguir en diferentes fases, como muestra la Figura 3.1, para facilitar su explicación y comprensión. Estas fases son altamente dependientes, de modo que modificaciones en fases iniciales conllevan variaciones en el resultado y prestaciones de las fases posteriores.

3.1. Conceptos generales

Una imagen es una representación 2D de una escena 3D. Las señales e imágenes de nuestro entorno son analógicas, es decir, existen en un tiempo continuo y toman un rango continuo de valores. Es necesario digitalizar las señales para poder ser almacenadas y manejadas por los ordenadores, definiéndose en un tiempo discreto y con un número discreto de valores. Tras este proceso de discretización obtendremos como resultado una imagen digital.

3.1.1. Tipos de imágenes

Una imagen digital es una representación bidimensional de una imagen mediante una matriz numérica. El píxel es la unidad mínima de visualización. Para el tratamiento digital de imágenes hay numerosos conceptos a tener en cuenta, a continuación se explican los relacionados con este proyecto.

En el caso de la visión humana, la señal en la que estamos interesados es la luz (onda electromagnética radiada). El ojo humano es sensible únicamente a la radiación comprendida en la banda definida por las longitudes de onda 350 a 780 nanometros, aproximadamente. Las propiedades principales de las ondas son luminancia y crominancia.

A) Luminancia

   La luminancia es la densidad de flujo luminoso que incide, atraviesa o emerge de una superficie siguiendo una dirección determinada. Alternativamente nosotros la definiremos como la intensidad luminosa que percibimos al reflejarse la onda en un objeto.

B) Crominancia

   La crominancia es la componente de la señal que contiene información del color. El color es un atributo de la percepción visual, asociado a las diferentes longitudes de onda del espectro visible. El color puede definirse a través de dos magnitudes, tinte y saturación. El tinte indica la tonalidad del color (relacionado con la magnitud de onda dominante); la saturación indica la pureza espectral (viveza del color).

   Los colores que percibimos por la vista corresponden a una gama continua de valores, pero en las imágenes digitales se usa una cantidad de colores (paleta de colores) finita. La profundidad de color n indica el número de bits usados para representar el color que asignaremos a un píxel y limita el número de colores de la paleta a 2^n.

Según los valores almacenados en los píxeles hay distintos tipos de imágenes. En este proyecto se han usado, fundamentalmente, los siguientes:

- Binaria

  Tiene únicamente dos posibles valores para cada píxel (un único bit), generalmente codificados como negro y blanco. El blanco se suele utilizar para codificar los píxeles asociados al objeto de la imagen y se denomina *foreground* o primer plano, mientras que el resto de la imagen es el *background* o fondo.

- Escala de grises

  Cada píxel tiene asociado un valor numérico correspondiente a un nivel de intensidad. Trabajaremos con intensidades representadas por 8 bits, disponiendo de 256 valores en el intervalo 0-255. Se les denomina también imágenes monocromo y representan distintos tipos de gris, variando desde negro cuando la intensidad es mínim hasta blanco cuando la intensidad es máxima. El aumento del número de bits repercute en el rango de niveles de intensidad a representar, y por tanto en la capacidad para representar suaves transiciones visuales.

- RGB (Red-Green-Blue)

  Modela cada color a partir de la combinación de energía luminosa asociada a tres longitudes de onda diferentes; rojo, verde y azul. De este modo se dispone de tres números (0-255) para representar la cantidad de energía asociada a las longitudes de onda roja, verde y azul, que combinaremos de manera aditiva para obtener el color del píxel. Es importante destacar que OpenCV usa la notación BGR (Blue-Green-Red): mismo espacio de color que RGB, pero con los canales de rojo y azul intercambiados.

Para un tamaño fijo de imagen, la resolución indica el grado de detalle y viene definido por el número de píxeles por pulgada, que guarda relación con el tamaño de la imagen (número de filas y columnas de la matriz numérica bidimensional).

El ojo humano es más sensible a los cambios producidos en los valores de luminancia que en los valores de crominancia, motivo por el que los métodos de reconocimiento de objetos suelen trabajar sobre imágenes en escala de grises. Para más información sobre la adquisición de imágenes, la visión humana y la caracterización de imágenes digitales, consúltense las referencias [13] [14] [15].

3.1.2. Transformaciones

Las imágenes, y en concreto la proyección de los objetos en la imagen capturada, pueden sufrir diversos cambios dependiendo del entorno. Entre estos cambios, caben destacar los siguientes:

- Iluminación: cambio en la luz que refleja un objeto.

- Oclusiones: ocultación parcial del objeto debida a la interposición de otros objetos en la trayectoria de visión.
- Trasformaciones geométricas: algunos ejemplos son traslación, escalado o rotación. Las explicaremos con más detalle a continuación.

Las transformaciones geométricas mapean los puntos $\mathbf{p} = (x, y, z)$ a puntos $\mathbf{p}' = (x', y', z') = f(x, y, z)$ en otro espacio mediante la función f. Estas transformaciones pueden ser simples, por ejemplo una traslación, o transformaciones (lineales y no lineales) más complejas. Hablaremos de transformaciones sobre figuras, en nuestro caso, la figura será la imagen.

En primer lugar consideraremos las *transformaciones euclídeas*, es decir, aquellas que no modifican el tamaño ni la forma de la figura, cambiando únicamente su posición en el espacio. Para definir la traslación y la rotación de un punto $\mathbf{p} = (x, y, z)$ en el espacio se usa la Ecuación 3.1.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.1}$$

Los valores t_x, t_y y t_z indican la translación en cada uno de los ejes, y los valores r_{ij} para $i, j = 1, 2, 3$, son los valores de la matriz de rotación que denominaremos R. Si rotamos sobre el eje x, usando la matriz de rotación R_x, se modificarán los valores de los ejes y y z. Las matrices de rotación se muestran en la Ecuación 3.2. Nótese que para la rotación en el eje x, el ángulo ϕ está en el plano definido por las otras dos variables, yz. De forma ánaloga definimos el resto de ángulos.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} R_z = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

En las *transformaciones euclídeas* la forma de la figura no varía, es decir, las líneas se transforman en líneas, los planos en planos, los círculos en círculos y las elipses en elipses. Únicamente cambia la posición y orientación de la figura. Las *transformaciones afines* son una generalización de las *transformaciones euclídeas* en las cuales las líneas son transformadas en líneas, pero los círculos pasan a ser elipses en el nuevo espacio. Las longitudes y los ángulos no se preservan en esta transformación, modificando la forma de la figura. La matriz general de una *transformación afín* viene definida por la Ecuación 3.3.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.3}$$

En esta ecuación vemos que las *transformaciones euclídeas* están incluidas si los valores de a_{ij} para $i, j = 1, 2, 3$ son los coeficientes de las matrices de rotación R_x, R_y o R_z. Por tanto, las *transformación euclídeas* son también *transformaciones afines*. En esta nueva transformación se incluyen las operaciones de escalado y elongación de la figura en una determinada dirección. El factor de escalado en cada dirección se representa mediante las variables s_x, s_y y s_z en la diagonal, como representa la matriz S de la Ecuación 3.4. La elongación la indicamos con la matriz E, donde los subíndices indican el plano en el que se realiza la elongación, y la variable x, y o z, el valor de la elongación en esa dirección.

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}, E = \begin{bmatrix} 1 & x_{xz} & x_{xy} \\ y_{yz} & 1 & y_{xy} \\ z_{yz} & z_{xz} & 1 \end{bmatrix} \tag{3.4}$$

A modo ilustrativo, a continuación se muestra un ejemplo para una transformación que represente una rotación y escalado en el eje x, una elongación en el plano yz, y una traslación genérica (véase la Ecuación 3.5).

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & t_x \\ e_y & \cos\phi & \sin\phi & t_y \\ e_z & -\sin\phi & \cos\phi & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \begin{array}{c} x' = s_x x + t_x \\ y' = e_y x + \cos\phi y + \sin\phi z + t_y \\ z' = e_z x - \sin\phi y + \cos\phi z + t_z \end{array} \tag{3.5}$$

Con la definición de la matriz de escalado S, la matriz de elongación E, y la matriz de rotación genérica $R = R_x R_y R_z$, podemos determinar una matriz $M = SRE$, tal que cualquier *transformación afín* es expresada por la Ecuación 3.6.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} & & & t_x \\ & M & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.6}$$

La *transformación proyectiva* es la transformación lineal más genérica, englobando las dos transformaciones anteriores. Son más genéricas porque la cuarta fila no tiene que ser estrictamente (0,0,0,1). En la *transformación proyectiva* las líneas son transformadas en líneas, pero a diferencia de los casos anteriores, no preservan necesariamente el paralelismo entre ellas. La *transformación proyectiva* viene definida por la Ecuación 3.7.

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{3.7}$$

La Figura 3.2 muestra diferentes transformaciones geométricas de la imagen *Lena*. Las dos primeras filas muestran algunas *transformaciones euclídeas*, la tercera fila *transformaciones afines* y la última fila *transformaciones proyectivas.*

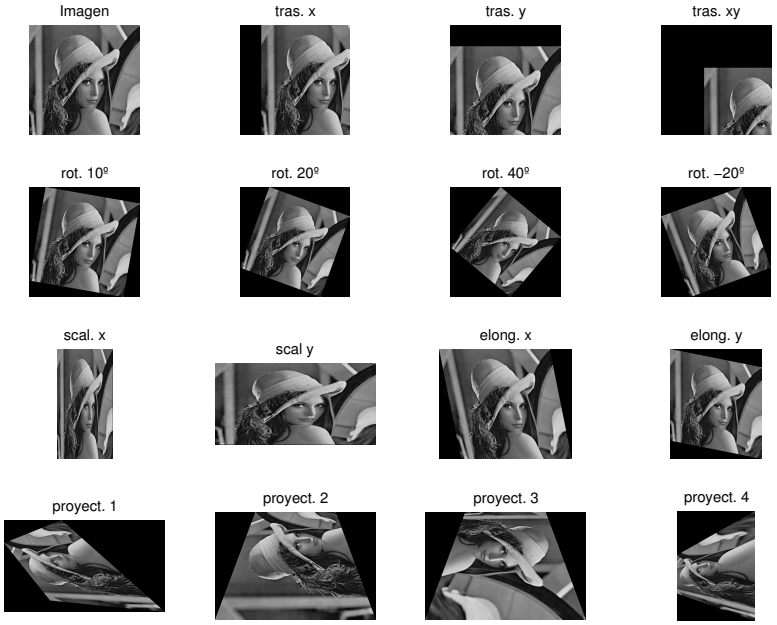


Figura 3.2: Imagen monocromo de *Lena* (512x512) con diferentes transformaciones geométricas. En los títulos de las figuras se han usado las siguientes equivalencias: tras=traslación, rot=rotación, scal=escalado, elong=elongación y proyect=proyectiva.

Para más información sobre las posibles transformaciones de una imagen, el método que emplean las cámaras para transformar el espacio real 3D al espacio 2D representado en una imagen y su transformación inversa, véase *Computational photography in image analysis and visualization* [16].

3.1.3. Histograma

Un *histograma* es la representación gráfica en forma de diagrama de barras del número de ocurrencias de los valores de una variable, donde la superficie de cada

barra es proporcional a la frecuencia de ocurrencia de los valores representados. Para una imagen digital en escala de grises, el histograma representa la frecuencia de ocurrencia de cada nivel de gris, como muestra la Figura 3.3.

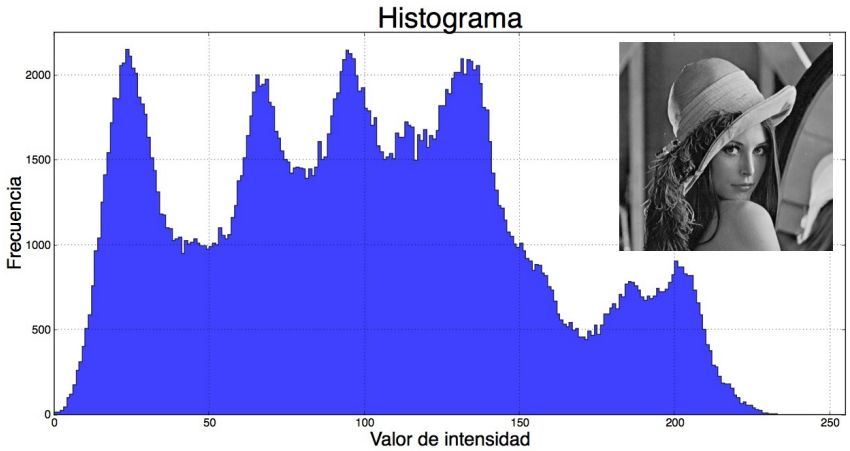


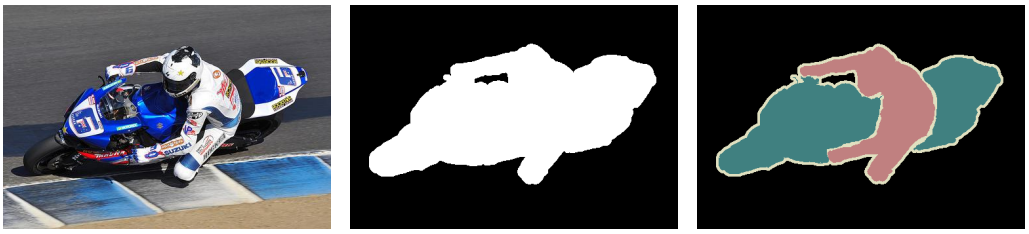
Figura 3.3: Imagen monocromo de *Lena* (512x512) y su respectivo histograma de niveles de intensidad.

El histograma corresponde a una reducción de dimensionalidad respecto a la imagen original. Esta reducción ocasiona una pérdida de información, principalmente espacial, y por tanto una imagen no puede ser reconstruída a partir de su histograma. Una imagen con un histograma compacto revela un uso pobre del rango de grises disponible, percibidiendo una falta de contraste visual en dicha imagen.

3.1.4. Segmentación

La segmentación de una imagen implica la división o separación de la misma en regiones con propiedades similares, con el objetivo de facilitar su posterior análisis. La luminancia es la característica básica para segmentar una imagen monocromo, y el color es la correspondiente para imágenes en color. La segmentación de una imagen es una de las etapas más complicadas en el proceso de visión por ordenador y no existe ningún esquema de segmentación ideal. Según las características de la imagen y el próposito final existen distintos tipos de segmentación. En la Figura 3.4 se muestran dos tipos de segmentación.

En este proyecto usaremos la segmentación para diferenciar entre el primer plano (*foreground*), que definirá los objetos de interés, y el fondo (*background*), que indicará todo lo que consideraremos irrelevante. De este modo, el resultado será una imagen binaria. A continuación explicaremos algunos de los esquemas de segmentación evaluados y usados en este proyecto.



(a) Original (b) Segmentación FG/BG (c) Segmentación semántica

Figura 3.4: Ejemplos de segmentación. La segmentación FG/BG o segmentación binaria representa todos los píxeles asociados al fondo como negro y los píxeles del primer plano como blanco. La segmentación semántica contiene diferentes clases dependiendo de si los píxeles pertenecen a la moto o al motorista. Datos tomados de *The PASCAL Visual Object Classes Challenge 2012 (VOC_{2012})* [17].

A) Umbralización simple (*Basic thresholding*)

Este es el método de segmentación más sencillo. Consiste en convertir una imagen monocromo I en una imagen binaria I_{th} mediante la transformación mostrada en la Ecuación 3.8. Si el valor del píxel es inferior a un determinado umbral, generalmente se le asigna $C_1 = 0$ y si es superior al umbral $C_2 = 255$.

$$I_{th}(x,y) = \begin{cases} C_1 & \text{si I(x,y)} \leq \text{umbral} \\ C_2 & \text{si I(x,y)} > \text{umbral} \end{cases} \tag{3.8}$$

Se utiliza cuando la imagen contiene un objeto de interés con luminancia razonablemente uniforme en un fondo con luminancia distinta. Si nos centramos en una imagen como la descrita, el histograma mostrará dos regiones claramente diferenciadas. La elección del umbral se hará de modo que separemos las dos regiones de la foma más clara posible.

B) Umbralización de Otsu (*Otsu's thresholding*)

Las imágenes son generalmente complejas, lo que dificulta la elección del umbral que diferencia primer plano del fondo. Este método se utiliza para determinar automáticamente el umbral basándose en la forma del histograma. Iteraremos para todos los posibles valores del umbral (posibles niveles de gris), y calcularemos la varianza de los niveles de intensidad de los píxeles a cada lado del umbral. Esta separación da lugar a la clase primer plano y a la clase fondo. El objetivo es encontrar el umbral óptimo que minimiza la varianza de niveles dentro de cada clase (varianza *intra-clase*), a la vez que maximiza la varianza entre clases (varianza *inter-clase*).

C) Crecimiento de regiones. Algoritmo *Watershed*

Este algoritmo establece una analogía entre una imagen monocromo y un relieve topográfico cuya altura viene definida por el valor de intensidad de los píxeles. Si sobre el relieve se vierte agua desde distintos puntos, ésta fluirá a lo largo del camino de mayor pendiente hasta que quede atrapada en un mínimo regional. El conjunto de fronteras de las distintas cuencas de una imagen constituye su segmentación por *watershed*.

En este tipo de algoritmos de crecimiento de regiones hay muchos factores a tener en cuenta. El punto crítico es la selección de los *puntos semilla*, siguiendo con la analogía, aquéllos sobre los que verteremos agua, pues los mínimos que localizaremos estarán directamente relacionados con esta elección. Su problema principal es la sobresegmentación, es decir, exceso de regiones distintas. Para solucionar este problema hemos de lidiar con la definición de criterios de propagación del agua y de combinación de cuencas. Es un algoritmo cuyo resultado depende de muchos parámetros (puntos semilla, criterio de propagación y criterio de combinación, entre otros) y que suele requerir de un preprocresado de la imagen para obtener buenos resultados.

D) GrabCut

GrabCut [18] es un método de segmentación iterativo basado en el algoritmo *Graph Cut* [19]. *GrabCut* extiende el algoritmo *Graph Cut* a imágenes en color y *trimaps* incompletos, cuyo significado explicaremos en su contexto más adelante. La inclusión de la información de color y el enfoque de aprendizaje iterativo aumenta su robustez. Este método es bastante complicado, por lo que daremos únicamente una intuición de su implementación.

En primer lugar describiremos las estructuras que se usan y que sirven como introducción a la terminología. Los parámetros almacenados en una estructura para cada píxel son los siguientes:

- Color: Es un valor RGB.
- Trimap: Segmentación inicial proporcionada por el usuario y cuyos valores pueden ser: *TrimapBackground*, *TrimapForeground* y *TrimapUnknown*.
- Matte: Segmentación dura determinada por el algoritmo, cuyos valores pueden ser *MatteBackground* y *MatteForeground*.
- Índice k: Número comprendido entre 1 y K, donde K es el número total de componentes de un *Gaussian Mixture Model (GMM)*. Indica a qué componente gaussiana K_k con $k = 1..K$ está asignado el píxel.

Para modelar *background* y *foreground*, *GrabCut* necesita dos *GMMs* de K componentes Gaussianas cada uno. Estos *GMMs* vendrán definidos por:

- μ: La media es una tupla de tres valores, del color RGB.
- $\mathbf{\Sigma^{-1}}$: La inversa de la matriz de covarianza de los valores de color, de dimensión 3x3.
- $det(\mathbf{\Sigma})$: El determinante de la matriz de covarianza, es decir, un número real.
- π: Peso que indica la fracción de píxeles, pertenecientes al *foreground* o *background*, asignados a cada componente gaussiana del *GMM*.

Los pasos principales del algoritmo son los siguientes:

1) Inicialización

   En primer lugar el usuario indica el *trimap* inicial mediante un rectángulo. Los píxeles incluidos dentro del rectángulo son marcados como *TrimapUnknown* y los del exterior como *TrimapBackground*. A partir del *trimap* introducido, el algoritmo realiza una segmentación inicial donde todos los píxeles marcados como *TrimapUnknown* se asignan a la clase *MatteForeground*, y los píxeles marcados como *TrimapBackground* se asignan a la clase *MatteBackground*. El *trimap* es la entrada del usuario y el *matte* es el resultado de la segmentación obtenida por el algoritmo *GrabCut*.

   A partir de esta segmentación, los dos *GMM* que modelan las clases *MatteForeground* y *MatteBackground* son inicializados. Para ello se dividen los píxeles de cada clase en K *clusters* y cada componente gaussiana es inicializada a partir de los colores de los píxeles asociados a cada *cluster*. Para cada una de las componentes gaussianas hemos de calcular su media, matriz de covarianza, determinante de la matriz de covarianza y peso.

2) Asignación de píxeles y estimación *GMMs*

   Cada píxel de la clase *MatteForeground* es asignado a la componente gaussiana con mayor probabilidad de producir su color en la *GMM* que modela el *foreground*. La asignación del píxel a la componente gaussiana K_k con mayor probabilidad se hace mediante la asignación del índice k de la estructura del píxel. Esta probabilidad se determina evaluando cada componente gaussiana con el color del píxel como entrada. El proceso se repite de forma análoga para los píxeles de la clase *MatteBackground*. Estas asignaciones estarán actualizando los K *clusters*. Tras la asignación de todos los píxeles, los dos *GMMs* se vuelven a estimar con los nuevos K *clusters*, donde cada *cluster* contiene los píxeles asignados a la componente gaussiana K_k.

3) Graph Cut

   A continuación se construye el grafo, teniendo un nodo por cada píxel, *nodo píxel*, y dos nodos especiales, el *nodo foreground* y el *nodo background*. Los

nodos son unidos mediante dos tipos de arcos. Los arcos denominados *N-links*, que conectan píxeles en la vecindad a 8, indican el coste de situar la frontera de segmentación entre dos píxeles vecinos. Cada *nodo píxel* tiene dos arcos *T-links*, uno que lo conecta al *nodo foreground* y otro que lo conecta al *nodo background*. Los pesos de estos arcos dependen del *trimap*. Si el usuario ha indicado que un píxel pertenece al *background* o al *foreground*, se le asigna el peso para forzar a ese píxel a ser asignado al nodo correspondiente. A los píxeles desconocidos se les asigna como pesos las probabilidades obtenidas para cada *GMM*. Posteriormente se aplica el algoritmo *Graph Cut*, que divide el grafo minimizando el peso total de los arcos por los que pasa, obteniendo un resultado como el de la Figura 3.5.

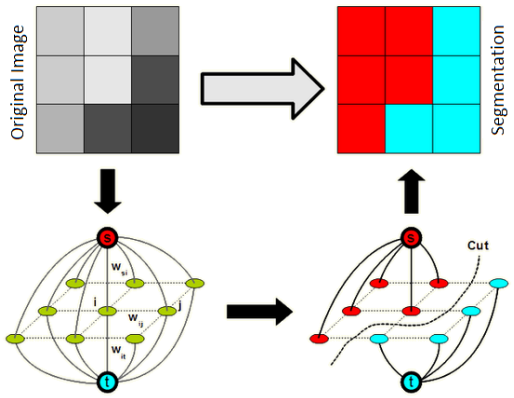


Figura 3.5: Ejemplo básico de segmentación basada en corte de grafos. Notese que en el ejemplo la imagen de entrada es monocromo y la vecindad es a 4. El *nodo foreground* es rojo y el *nodo background* es cyan. Imagen tomada de *Graph Cut based image segmentation in fluorescence microscopy* [20].

4) Proceso iterativo

   Los pasos de asignación de píxeles, estimación de *GMMs* y *Graph Cut* se repiten hasta que converge la clasificación de los píxeles.

Tras finalizar este proceso obtendremos dos conjuntos de píxeles, los asignados al *foreground* y los asignados al *background*. Aunque las principales etapas de este algoritmo son las indicadas anteriormente, algunos investigadores han añadido pasos para matizar los bordes de las fronteras FG/BG o recibir más información del usuario para paliar errores de clasificación, entre otros. Para más información sobre *GrabCut* y su aplicación en segmentación semántica, consulte las siguientes referencias [21] [22].

3.1.5. Sustracción del fondo

El objetivo es extraer el primer plano (*foreground*) de una imagen, para su posterior procesado. La sustracción de fondo se usa generalmente si la imagen en cuestión es parte de un vídeo. Es un enfoque muy usado para detección de objetos/personas en movimiento en vídeos tomados por cámaras estáticas [23].

Figura 3.6: Ejemplo de sustracción del fondo. De izquierda a derecha se muestra la imagen del fondo, la imagen con el primer plano incluido, estimación del fondo, estimación de las sombras, estimación del primer plano y resultado final aplicando *GrabCut*. Imagen tomada de *GrabCut background subtraction, a C implementation* [24].

Hay diversos métodos para modelar el fondo; a continuación explicaremos algunos de los explorados en este proyecto. Para ver una comparativa consúltese [25].

A) Diferencia básica

Este método de modelado del fondo está basado en el historial reciente del píxel, $\mathbf{p} = (x, y)$. Consideremos dos imágenes consecutivas $V(\mathbf{p}, t)$ y $V(\mathbf{p}, t+1)$ tomadas por una cámara de vídeo. En este enfoque se asume que la primera imagen corresponde al fondo y es sustraída de la segunda imagen, tal como se indica en la Ecuación 3.9. Si la diferencia supera un umbral, se considera que el píxel pertenece al primer plano y, como se indica en la Ecuación 3.10, se le asigna un valor C_2. En caso contrario se le asigna el valor C_1.

$$BS(\mathbf{p}) = |V(\mathbf{p}, t+1) - V(\mathbf{p}, t)| \tag{3.9}$$

$$Mask_{BD}(\mathbf{p}) = \begin{cases} C_1 & \text{si BS}(\mathbf{p}) \leq \text{umbral} \\ C_2 & \text{si BS}(\mathbf{p}) > \text{umbral} \end{cases} \tag{3.10}$$

Este enfoque sólo detectará los píxeles correspondiente al primer plano si siempre están en movimiento, y su precisión depende directamente de la velocidad de movimiento de los objetos en la escena. Hay modificaciones de

esta técnica que, en lugar de tomar solo dos imágenes consecutivas, estiman la imagen correspondiente al fondo ponderando el valor de intensidad de los píxeles de una secuencia de imágenes más larga, por ejemplo las veinte imágenes anteriores a la actual. Este método viene descrito por la Ecuación 3.11 y se denomina *mean filter*, donde N indica el número de imágenes precedentes que se usarán para calcular la media. Posteriormente se tomará la decisión de forma similar a la explicada anteriormente para la diferencia básica, como se indica en la Ecuación 3.12.

$$B(\mathbf{p}) = \frac{1}{N}\sum_{i=1}^{N} V(\mathbf{p}, t_i) \tag{3.11}$$

$$Mask_{MD}(\mathbf{p}) = \begin{cases} C_1 & \text{si } |\text{V}(\mathbf{p},\text{t})\text{-B}(\mathbf{p})| \leq \text{umbral} \\ C_2 & \text{si } |\text{V}(\mathbf{p},\text{t})\text{-B}(\mathbf{p})| > \text{umbral} \end{cases} \tag{3.12}$$

Si el píxel es clasificado como primer plano, será ignorado en la estimación del fondo para la imagen en el siguiente instante de tiempo. De este modo se previene que el modelo del fondo sea contaminado por píxeles que no pertenecen al mismo.

B) Combinación de Gaussianas

El modelado denominado *mean filter* tiene alta velocidad de computación, pero falla en entornos no controlados. Los problemas más comunes son cambios en iluminación y desorden del fondo que se produce en las escenas al aire libre con el paso del tiempo. Este problema se afronta usando un modelo más complejo, denominado *Mixture of Gaussians (MoG)*, que permite representar fondos multimodales y ajustar sus propios parámetros de forma adaptativa.

En este enfoque se asume que la intensidad de cada píxel a lo largo del tiempo se puede modelar mediante una combinación de Gaussianas. En imágenes monocromo la intensidad es un escalar, mientras que en imágenes en color corresponde a un vector. En cada instante de tiempo t se conoce el historial de valores de cada píxel $\mathbf{p} = (x_0, y_0)$ a lo largo del tiempo, i.e. $\{x_1, ..., x_t\} = \{I(\mathbf{p}, i) : 1 < i < t\}$, que definiremos como $\mathbf{x}$. Como indica la Ecuación 3.13, este historial se modela con una combinación de K distribuciones gaussianas , donde w_k son los pesos y $\mathcal{N}(\mathbf{x}; \mu_k, \sigma_k)$ es la distribución normal de media μ_k y matriz de covarianza $\mathbf{\Sigma}_k = \sigma_k \mathbf{I}$ ($\mathbf{I}$ denota la matriz identidad). Esta asunción de independencia entre las dimensiones de cada Gaussiana fue realizada por sus creadores [26].

$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}; \mu_k, \sigma_k) \tag{3.13}$$

Se comprueba si cada píxel x_t dista de la media μ_k de cada Gaussiana un valor inferior a τ (valor umbral), en cuyo caso (*match*) el valor del píxel x_t se usará para actualizar los parámetros w_k, μ_k y σ_k. Las expresiones de actualización de los parámetros pueden consultarse en [26] y no serán descritas por brevedad. Únicamente destacaremos la existencia de un párametro α que representa la tasa de aprendizaje, es decir, la importancia que daremos al nuevo píxel y por tanto la velocidad de actualización de la componente gaussiana. Si ninguna de las K_k componentes gaussianas cumplen la condición de *matching*, la menos probable será reemplazada por una nueva distribución con alta varianza, bajo peso y donde la media será el valor del píxel actual x_t.

A continuación se determina qué componentes gaussianas son originadas por píxeles que pertenecen al fondo, para lo cual se ordenan los cocientes w_k/σ_k. La *función densidad de probabilidad* del fondo se modela utilizando las primeras B componentes que cumplen la Ecuación 3.14, donde T indica la porción mínima de píxeles a modelar como fondo.

$$B = argmin_b(\sum_{k=1}^{b} w_k > T) \tag{3.14}$$

Tras lo anterior hemos de buscar la componente gaussiana que mejor modela el valor actual del píxel, para lo cual se determina el valor de k que maximiza $P(k|x_t, \mathbf{\Phi})$, donde k hace referencia a la k-ésima componente gaussiana, x_t es el valor actual del píxel y $\mathbf{\Phi} = \{w_i, \mu_i, \sigma_i\}$ para $i = 1..b$ es el conjunto de parámetros del modelo. En lugar de aplicar un método analítico se usa una aproximación [27], de modo que si el valor actual del píxel dista de la media de la gaussiana $k-$ésima menos de $2{,}5\sigma_k$, el píxel se considerará como fondo y se le asignará el valor 1. En caso de que yazca en dicha region para más de una componente gaussiana, se selecciona la primera. Este proceso se realiza para todos los píxeles, obteniendo una imagen binaria (0,1) según el píxel pertenezca al primer plano o al fondo, respectivamente.

El proceso anterior se realiza para cada píxel a lo largo del tiempo, y permite identificar los píxeles que pertenecen al fondo a la vez que se actualiza el modelo del fondo. Los píxeles del primer plano son agrupados usando un análisis de componentes conexas en 2D. Un ejemplo de su uso para estimar patrones *Bayern-Pattern*[1], es analizado por Jae K., Ho Gi, Gen Li y Jaihie Kim [28]. Para más información sobre *MoG* se pueden consultar las referencias [29] [30].

[1]Tipo de patrón usado en los sensores de las cámaras. Patrón formado por repeticiones de un cuadrado de 2x2 ([G B; R G]). G=verde, B=azul y R=rojo.

3.2. Estado del arte

Tras adquirir la imagen y realizar, si se considera necesario, el preprocesamiento (realce, segmentación y sustracción de fondo) se deben extraer las características más relevantes para el reconocimiento de objetos. Debido a que las imágenes tienen gran cantidad de datos no relevantes, usamos *detectores* para localizar los píxeles de interés, posiciones *(x,y)* de la imagen. Posteriormente se utilizarán *descriptores* para extraer las características de los puntos de interés detectados previamente.

A) Detectores

   Los detectores seleccionan los puntos más significativos de una imagen, a los que denominaremos *puntos de interés*. Estos puntos de interés se encuentran en zonas de la imagen con mucho contraste, en los bordes de objetos y en las esquinas. Es importante destacar que estos puntos, por sí solos, no ofrecen mucha información, por lo que se define una región de interés en torno a los mismos.

B) Descriptores

   Los descriptores se aplican a las regiones de interés de cada punto, para obtener los parámetros que mejor describen sus características físicas. De este modo se pueden describir propiedades tales como textura, color o forma, entre otras.

Como se indicó anteriormente, son numerosas las posibles transformaciones entre la proyección del objeto en dos imágenes, siendo tanto mejores los resultados cuanto más robustos sean *detector* y *descriptor* a estas transformaciones.

3.2.1. Detectores y descriptores. Introducción

La literatura relacionada con la detección y descripción de características en imágenes es muy amplia, y nos resulta imposible describir en detalle cada una de las contribuciones realizadas. El objetivo general de esta subsección es introducir al lector algunas de las ideas que han sido propuestas.

En principio se usaron *descriptores globales* que describían el contenido completo de la imagen basándose en el color o la intensidad. Para superar las limitaciones y mejorar las prestaciones de estos descriptores, se ideó un enfoque basado en segmentar la imagen en diferentes regiones, para posteriormente describirlas. Es así como aparecen los *descriptores locales* que hacen uso de los detectores para determinar las regiones de interés, y que serán la parte central de este proyecto.

Con el paso del tiempo se fueron desarrollando técnicas para solventar problemas originados por transformaciones en la imagen, entre ellos cabe destacar el

escalado, la iluminación, las tranformaciones geométricas y el ruido. En la actualidad existen numerosos métodos de detección y descripción de puntos de interés, y muchos de ellos pueden ser combinados entre sí. A continuación explicaremos brevemente algunos de ellos.

1. FAST

En el *detector* FAST (*Features from Acelerated Segment Test*) [31] se inspecionan los valores de intensidad de los píxeles en un círculo de radio r alrededor del píxel candidato $\mathbf{p} = (x, y)$, como muestra la Figura 3.7a. Un píxel del círculo es considerado *"bright"* si su intensidad es al menos t (valor umbral) unidades superior a la intensidad del píxel candidato $\mathbf{p}$, y *"dark"* si su intensidad es al menos t unidades inferior. El píxel candidato $\mathbf{p}$ es considerado punto de interés si al menos un arco de longitud n píxeles *"bright"* o *"dark"* es encontrado en el círculo. En la implementación original se usan los valores de $r = 3$ y $n = 9$.

2. STAR

En el desarrollo de STAR [32] los autores tenían como objetivo crear un *detector* multi-escala con resolución espacial completa, ya que en otros detectores (como SIFT) el submuestreo realizado a la imagen de entrada afecta a la precisión de la localización del punto de interés. Este detector usa una aproximación del *Laplacian of Gaussian (LoG)* y recibe su nombre por la forma de la máscara usada (véase la Figura 3.7b). Esta máscara preserva invarianza rotacional y habilita el uso de imágenes integrales para realizar los cálculos de forma eficiente. El espacio escalado se crea sin usar interpolación, aplicando máscaras de diferentes tamaño.

3. BRIEF

El *descriptor* BRIEF (*Binary Robust Independent Elementary Features*) [33] usa *strings* binarios para describir las características de los puntos de interés y su posterior emparejamiento. Esto habilita el uso de la distancia *Hamming* [2] para calcular la similitud entre dos descriptores, siendo computada de forma más eficiente que la comúnmente usada distancia Euclídea. Debido a la alta sensibilidad de BRIEF al ruido, la imagen se difumina con un filtro de media.

El valor de cada *bit* que define el descriptor depende del resultado de la comparación entre los valores de intensidad de dos píxeles pertenecientes a un segmento de la imagen centrado en el punto de interes a describir. El bit será 1 si el valor de intensidad del primer punto del par es mayor que el

[2]La distancia *Hamming* entre dos *strings* de igual longitud es el número total de posiciones en las que los símbolos correspondientes son distintos. Mide el mínimo número de sustituciones necesarias para convertir un *string* en el otro.

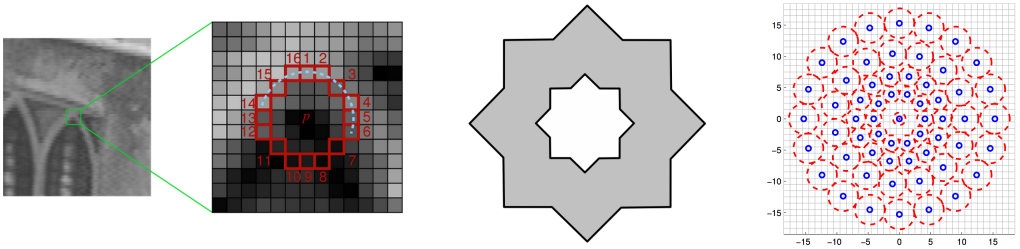
valor de intensidad del segundo punto, y 0 en caso contrario. El descriptor propuesto tiene una longitud de 512 bits calculados en una región de 48x48 píxeles. La forma básica de BRIEF no es invariante a rotaciones.

4. BRISK

El *detector* y *descriptor* BRISK (*Binary Robust Invariant Scalable Keypoints*) [34] hace uso del espacio escalado para la detección de los puntos de interés. La localización y la escala de cada punto de interés se obtiene ajustando una función cuadrática en el dominio continuo. La idea es usar un patrón para muestrear la vecindad del punto de interés, como muestra la Figura 3.7c, obteniendo valores de grises. Posteriormente se comparan pares de dichos valores, asignando al bit el valor 1 si la intensidad del primer elemento es mayor, y 0 en caso contrario. El emparejamiento se realiza de forma muy eficiente dada la naturaleza binaria del descriptor.

5. ORB

Este *detector* y *descriptor* es propuesto como una alternativa computacionalmente eficiente a SIFT y SURF, con una robustez y precisión de emparejamientos similar a la de SIFT y mejor que SURF. Está basado en el *detector* FAST y el *descriptor* BRIEF. ORB (*Oriented FAST and Rotated BRIEF*) [35] añade una componente de orientación a FAST (oFAST) y calcula los descriptores BRIEF de forma más eficiente, mejorando su comportamiento frente a rotaciones (rBRIEF).



(a) *detector* FAST (b) *detector* STAR (c) *descriptor* BRISK

Figura 3.7: (a) Ejemplo de detección FAST, (b) Máscara usada en SURF y (c) Patrón de muestreo usado en BRISK. Imágenes tomadas de la pagina oficial de FAST (`http://www.edwardrosten.com/work/fast.html`), [32] y [34] respectivamente.

Para más información sobre la evolución de los detectores y descriptores, su implementación y clasificación, consúltese [36]. Para una evaluación de los esquemas más modernos en el ámbito de la robótica, véase [37]. Los métodos que

explicaremos en este proyecto son los descriptores SIFT [38] y los descriptores SURF [39].

3.2.2. Descriptor SIFT

Scale Invariant Feature Transform (SIFT) es un algoritmo para detectar puntos de interés y describir características locales asociadas a esos puntos en imágenes. Este algoritmo fue publicado por David Lowe [38] y ha sido estudiado por muchos investigadores para mejorar su rendimiento y robustez, dando lugar a variantes como PCA-SIFT [40] o ASIFT [41]. Entre las posibles aplicaciones cabe destacar modelado 3D, reconocimiento de objetos, reconocimiento de gestos o *robot mapping*.

En nuestro caso particular, para conseguir un reconocimiento de objetos fidedigno, es importante que las características extraídas sean robustas a posibles cambios en la imagen de entrada. Algunos de estos posibles cambios fueron explicados en la Subsección 3.1.2. Los puntos de interés que son detectados por SIFT corresponden a zonas de la imagen con alto contraste, como bordes o esquinas de objetos. Otra propiedad de los puntos de interés a tener en cuenta es que su posición relativa en la escena no varíe dependiendo de la vista de la escena proyectada en la imagen.

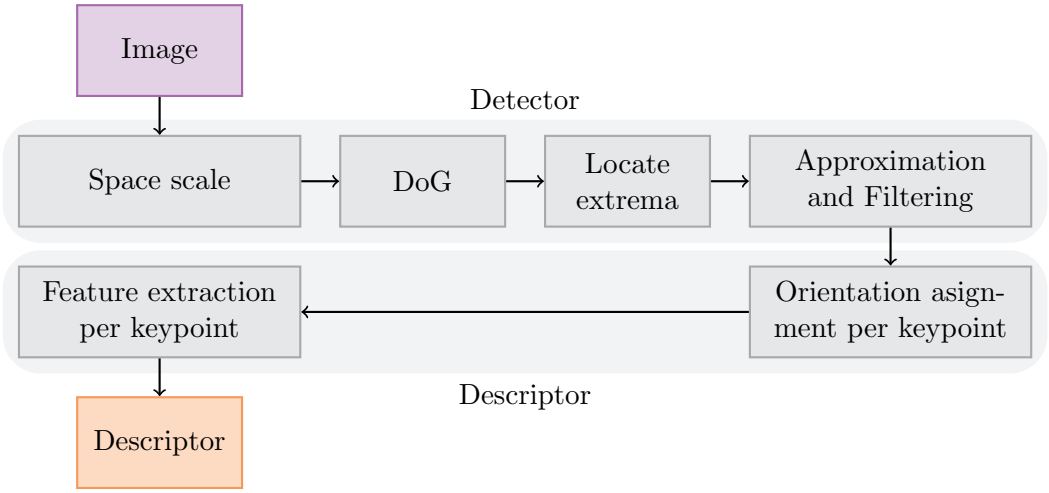


Figura 3.8: Diagrama con las etapas seguidas para obtener los descriptores SIFT de una imagen. El proceso se divide en dos fases: detección y descripción.

A continuación se explica el propósito y detalles de implementación para cada bloque del diagrama de la Figura 3.8.

A) Espacio escalado

En la primera etapa de este algoritmo, la imagen $I(x, y)$ es convolucionada de forma iterativa con un filtro espacial gaussiano $G(x, y, \sigma)$ definido por la Ecuación 3.15.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{3.15}$$

En la convolución de la imagen original $I(x, y)$ con el filtro espacial gaussiano $G(x, y, \sigma)$, incrementar σ tiene el efecto de filtrar las frecuencias altas, manteniendo las frecuencias bajas. Esto provocará un suavizado (*blurring*) en la imagen resultante, denominada $L(x, y, \sigma)$ y definida por la Ecuación 3.16.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.16}$$

El objetivo es crear un espacio escalado como el que se muestra en la Figura 3.9 para la imagen de *Lena*, que denominaremos *scale-space pyramid*. No todas las implementaciones de SIFT crean el espacio escalado de la misma manera. Nosotros nos centraremos en la definida por Lowe [38].

Figura 3.9: Ejemplo del espacio escalado usado en SIFT para la imagen de *Lena*.

El factor k de incremento del escalado viene definido por $k = 2^{1/s}$, donde $s+3$ es el número total de imágenes por octava. En la implementación original de SIFT, Lowe usa 4 octavas con 5 imágenes en cada una, como indica de forma genérica la Tabla 3.1. Además, la imagen es diezmada por un factor 2 cuando el parámetro σ duplica su valor, reduciendo significativamente el tiempo de computación.

	Octavas			
Escala	1	2	3	4
blur 1	$k\sigma = \sigma_1$	$2\sigma_1$	$4\sigma_1$	$8\sigma_1$
blur 2	$k^2\sigma = \sigma_2$	$2\sigma_2$	$4\sigma_2$	$8\sigma_2$
blur 3	$k^3\sigma = \sigma_3$	$2\sigma_3$	$4\sigma_3$	$8\sigma_3$
blur 4	$k^4\sigma = \sigma_4$	$2\sigma_4$	$4\sigma_4$	$8\sigma_4$
blur 5	$k^5\sigma = \sigma_5$	$2\sigma_5$	$4\sigma_5$	$8\sigma_5$

Tabla 3.1: Tabla general de los valores de escalado para 4 octavas y 5 imágenes por octava.

Como se indica en la Tabla 3.2, una octava tiene en común con la escala anterior la mitad de dimensiones de escalado. Este solapamiento favorece que el diezmado de la imagen no provoque grandes pérdidas en la precisión.

	Octavas			
Escala	1	2	3	4
blur 1	0.707107	1.414214	2.828427	5.656854
blur 2	1.000000	2.000000	4.000000	8.000000
blur 3	1.414214	2.828427	5.656854	11.313708
blur 4	2.000000	4.000000	8.000000	16.000000
blur 5	2.828427	5.656854	11.313708	22.627417

Tabla 3.2: Ejemplo numérico de los valores de escalado para 4 octavas con 5 imágenes cada una, donde el valor de σ inicial es $\sigma = 1/2$. En color están representados los valores de escalado en común entre diferentes octavas.

B) Diferencia de Gaussianas (DoG)

El Laplaciano de la Gaussiana, del inglés *Laplace of Gaussian (LoG)*, es un tipo de filtro espacial que detecta los bordes de foma robusta. Su cálculo es muy costoso, por lo que se usa una aproximación basada en la diferencia de gaussianas $D(x, y, \sigma)$ indicada en la Ecuación 3.17. La validez de esta aproximación fue demostrada por Lindeberg [42].

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{3.17}$$

La aproximación $D(x, y, \sigma)$ es la resta de dos escalas adyacentes de una misma octava, calculadas como se ilustra en la Figura 3.10.

Al finalizar este proceso se obtendrá un conjunto de imágenes al que se denomina *DoG-pyramid* (pirámide azul de la Figura 3.10), que contiene imágenes similares a las representadas en la Figura 3.11.

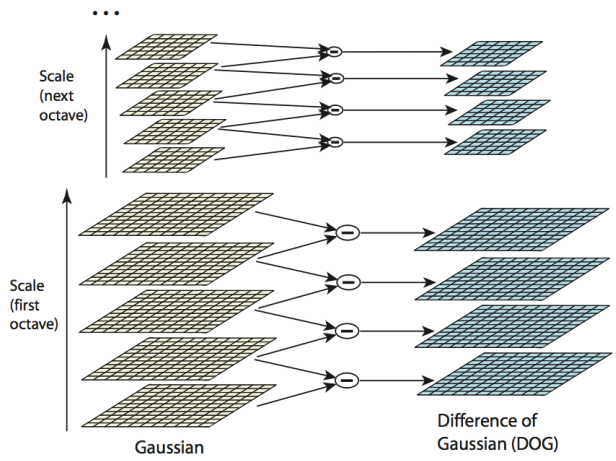


Figura 3.10: Ejemplo gráfico del cálculo de la Diferencia de Gaussianas (DoG). Esta diferencia se realiza para obtener la aproximación LoG. Las imágenes son reducidas por un factor de 2 para cada octava. Imagen tomada de [38].

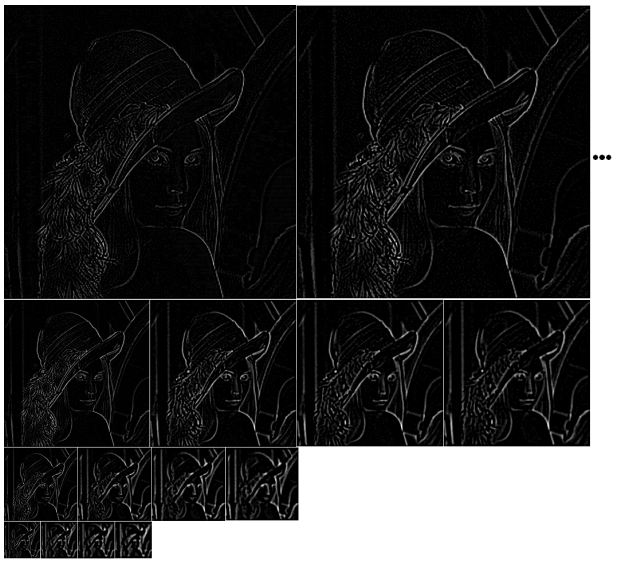


Figura 3.11: Ejemplo de las Diferencias de Gaussianas (DoG) usadas en SIFT para la imagen de *Lena*.

C) Localización de los extremos de DoG

Tras obtener la *DoG-pyramid* con el conjunto de imágenes a diferentes escalas, hemos de encontrar los *extremos* que potencialmente serán considerados *puntos de interés*. Para localizar los extremos, la intensidad de cada píxel $D(x, y, \sigma)$ es comparada con todos sus vecinos, es decir, con los 8 vecinos del mismo nivel, los 9 vecinos del nivel inferior y los 9 vecinos del nivel superior. Esta comparación se muestra en la Figura 3.12. Sólo si el valor del punto com-

parado es mayor (o menor) que sus 26 vecinos, éste será considerado como extremo.

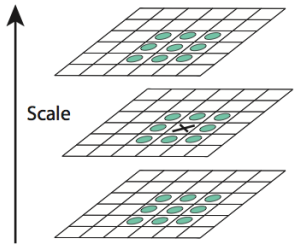


Figura 3.12: Localización de extremos en Diferencias de Gaussianas (DoG) comparando un determinado punto, representado por la cruz, con sus 26 píxeles vecinos, representados por círculos. Imagen tomada de [38].

D) Aproximación de puntos de interés y filtrado

Las posiciones de los extremos localizados en el paso anterior son obtenidas a diferentes escalas de la imagen original. Estas posiciones pueden estar definidas por valores decimales, pero en la representación de una imagen en forma de matriz sólo es posible acceder a posiciones definidas por números enteros. En las recientes implementaciones de SIFT se propone un método de interpolación del punto de interés aproximando una parábola 3D a los datos cercanos al extremo, que denominaremos *puntos muestra*. La aproximación, ilustrada en la Figura 3.13, se explica a continuación.

Para realizar la interpolación se usa la serie de expansión de Taylor cuadrática, Ecuación 3.18, de la diferencia de Gaussianas $D(x, y, \sigma)$. D y sus derivadas son evaluadas en los *punto muestra* y $\mathbf{x} = (x, y, \sigma)$ es el desplazamiento de esos *puntos muestra*.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial^2 \mathbf{x}}\mathbf{x} \tag{3.18}$$

La localización final del extremo $\hat{\mathbf{x}}$ coincidirá con uno de los *puntos muestra* (verde en la representación de la Figura 3.13), y se halla derivando la expansión de Taylor respecto de $\mathbf{x} = (x, y, \sigma)$ e igualando dicha derivada a 0. Centraremos el eje de coordenadas en el *punto muestra* actual, definiendo el *offset* $\hat{\mathbf{x}}$ como la distancia entre el *punto muestra* y el extremo. La expresión para la obtención del *offset* $\hat{\mathbf{x}}$ viene definida por la Ecuación 3.19.

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial^2 \mathbf{x}}\frac{\partial D}{\partial \mathbf{x}} \tag{3.19}$$

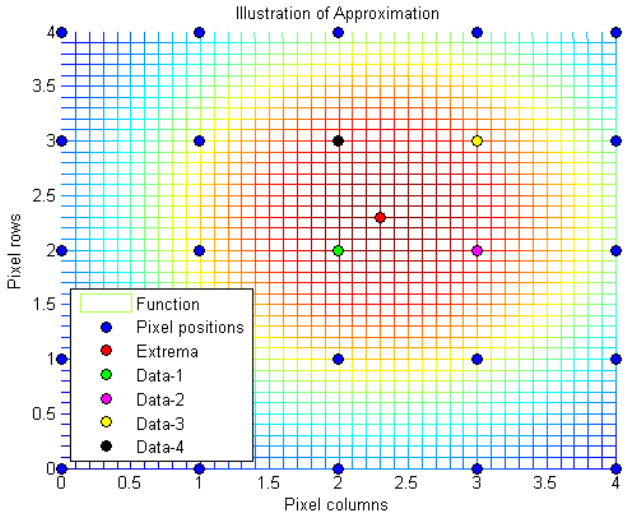


Figura 3.13: Aproximación del extremo con respecto a sus *puntos muestra*. La función representada es en realidad una Gaussiana para hacer la ilustración más intuitiva.

Si el *offset* $\hat{\mathbf{x}}$ es mayor que 0.5 en cualquier dimensión, el extremo yace más cercano a otro *punto muestra* y el proceso se repite para un *punto muestra* nuevo. El valor final del *offset*, $\hat{\mathbf{x}}$, se añade al extremo para interpolar la nueva localización del extremo según se indica en la Ecuación 3.20, donde $\hat{\mathbf{x}}_I$ es el punto interpolado, $\mathbf{x}_E$ el extremo y $\hat{\mathbf{x}}$ el *offset*.

$$\hat{\mathbf{x}}_I = \mathbf{x}_E + \hat{\mathbf{x}} \tag{3.20}$$

Debido a las escalas utilizadas para localizar los extremos, se dispone de un alto número de ellos, algunos de los cuales son inestables. El siguiente paso es descartar extremos de bajo contraste o con alta variación sólo en una dirección, y que por tanto son muy sensibles al ruido.

El valor de la función en el extremo, $D(\hat{\mathbf{x}}_I)$ de la Ecuación 3.21, es útil para descartar extremos inestables debido al bajo contraste. En los experimentos realizados por Lowe, todos los extremos con valor $|D(\hat{\mathbf{x}}_I)| \leq 0{,}03$ son descartados.

$$D(\hat{\mathbf{x}}_I) = D + \frac{1}{2}\frac{\partial^2 D^T}{\partial^2 \mathbf{x}}\hat{\mathbf{x}}_I \tag{3.21}$$

En segundo lugar se descartan los extremos situados a lo largo de bordes. De forma intuitiva los bordes tendrán una curvatura (variación o gradiente) pequeña a lo largo del mismo, y una curvatura alta si cruzamos a través de

él. Si $\hat{\mathbf{x}}_I$ es una esquina, este punto tendrá alta curvatura en ambas direcciones. Para descartar los extremos situados en bordes, se calcula el *ratio* entre curvaturas y se descartan aquéllos de alto *ratio*. Estas curvaturas pueden ser halladas aproximando la matriz Hessiana, $\mathbf{H}$ mediante diferencias entre píxeles vecinos en la localización y escala del extremo, $\hat{\mathbf{x}}_I = (x, y, \sigma)$. La matriz Hessiana viene definida por la Ecuación 3.22; al ser simétrica $D_{xy} = D_{yx}$.

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{3.22}$$

Hallaremos la traza y el determinante de $\mathbf{H}$ como se indica en la Ecuación 3.23, donde se ha definido $\alpha = r\beta$, siendo r el *ratio*.

$$\begin{aligned} Tr(\mathbf{H}) &= D_{xx} + D_{yy} = \alpha + \beta \\ Det(\mathbf{H}) &= D_{xx}D_{yy} - D_{xy}^2 = \alpha\beta \end{aligned} \tag{3.23}$$

Tomaremos la decisión mediante la Ecuación 3.24. En los experimentos de Lowe, el valor usado para el *ratio* es r=10 (elimina extremos de *ratio* entre curvaturas mayor que 10).

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r} \tag{3.24}$$

Los extremos no descartados en ninguna de las dos etapas de filtrado serán considerados *puntos de interés*.

E) Asignación de la orientación dominante a los puntos de interés

Las orientaciones han de ser asignadas a los *puntos de interés* para que el píxel sea descrito de forma relativa a la orientación, consiguiendo así invarianza frente a rotaciones. Para un determinado píxel, los parámetros del gradiente son la *magnitud* $m(x, y)$ y la *orientación* $\theta(x, y)$ y se aproximan mediante diferencias de valores en píxeles próximos, tal y como se indica en la Ecuación 3.25.

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \\ \theta(x,y) &= tan^{-1}\left(L(x,y+1) - L(x,y-1)\right) / \left(L(x+1,y) - L(x-1,y)\right) \end{aligned} \tag{3.25}$$

Las magnitudes y orientaciones de los píxeles con la misma escala que la del *punto de interés,* $\mathbf{k} = (x, y, \sigma)$, que están contenidos en una Gaussiana de desviación típica 1,5σ son usados para calcular un histograma (véase Figura 3.14) con 36 contenedores o *bins* (uno por cada 10 grados). Cada píxel contribuye al histograma de su respectiva orientación con su magnitud, ponderada por la ventana gaussiana centrada en el *punto de interés*.

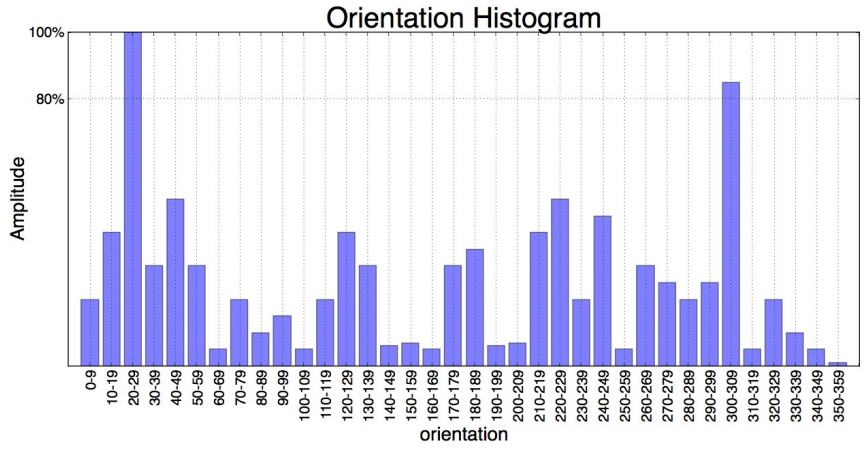


Figura 3.14: Histograma de orientaciones con 36 contenedores (uno por cada 10 grados). La altura de un contenedor representa la suma de magnitudes, ponderada por la ventana gaussiana centrada en **k**, de todos los píxeles cuya orientación corresponde con dicho contenedor.

Una vez completado el histograma, la orientación más dominante, representada por el contenedor más alto, es asignada al *punto de interés*, definido ahora por $\mathbf{k} = (x, y, \sigma, \theta_D)$, donde θ_D representa la orientación dominante. El punto de interés **k** es replicado para todos los contenedores cuya amplitud es al menos un 80 por ciento la máxima altura del histograma, asignándoles la orientación correspondiente a su contenedor (*bin*). Para cada orientación dominante se ajusta una parábola a los tres valores del histograma más próximos, interpolando la orientación para mayor precisión.

Tras finalizar este paso se dispone del *punto de interés* definido por $\mathbf{k} = (x, y, \sigma, \theta_D)$, donde (x, y) es la localización del punto, σ representa la escala y θ_D es la orientación principal de su entorno.

F) Cálculo de los descriptores

En los pasos anteriores hemos seleccionado los puntos de interés a diferentes escalas y les hemos asignado sus orientaciones principales. Esto garantiza cierta robustez frente a desplazamiento, escalado y rotación de la imagen. Ahora queremos definir una serie de valores que describan el punto de interés de forma única y distintiva frente al variaciones como iluminación o cambio de perspectiva 3D.

Se desea describir la región a la que hace referencia cada punto de interés, definido como $\mathbf{k} = (x, y, \sigma, \theta_D)$. Para ello se selecciona una región de 16x16 píxeles, que se divide en ventanas de 4x4 píxeles, para cada una de las cuales se crea su correspondiente histograma de orientación (como en el paso anterior).

Estos histogramas están definidos por 8 contenedores (uno por 45 grados) como se muestra en la Figura 3.15.

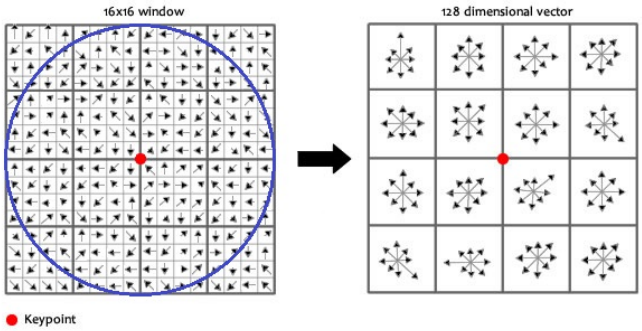


Figura 3.15: Histograma de orientaciones para la descripción de la región entorno al punto de interés. Las magnitudes son ponderadas por una gaussiana (círculo azul) centrada en el *punto de interés* y añadidas a su correspondiente contenedor. Cada histograma tiene 8 contenedores (orientaciones) cuya magnitud total es la longitud de la flecha.

A continuación se crea un vector con todos los valores de los histogramas, es decir, un vector de longitud 4x4 (número de ventanas) con 8 orientaciones (contenedores) por ventana, proporcionando un total de 128 valores. A este vector le denominaremos *descriptor*. La orientación principal de la región en torno al *punto de interés*, hallada en el paso anterior, se sustrae a los valores del *descriptor*. Finalmente, el *descriptor* es normalizado para hacerlo invariante a cambios de iluminación. Para eliminar efectos no lineales como los originados por la saturación de la cámara, Lowe reduce la influencia de las magnitudes grandes saturando los componentes del *descriptor* para que no haya valores mayores que 0.2 (determinado experimentalmente). Posteriormente se vuelve a normalizar para obtener un *descriptor* normalizado.

3.2.3. Descriptor SURF

Speeded-Up Robust Features (SURF) es un detector de características robusto presentado por Herbert Bay [39] y parcialmente basado en el anteriormente explicado SIFT [38]. Puede ser usado en ámbitos de visión por ordenador similares a los ya citados, como reconocimiento de objetos o reconstrucción 3D. La versión standard de SURF es varias veces más rápida que SIFT y por ello su uso en aplicaciones de tiempo real es muy extendida, como es el caso de seguimiento de objetos en vídeo.

Si se compara el esquema de SURF con el presentado anteriormente para SIFT, se observa que las diferencias más significativas son las siguientes:

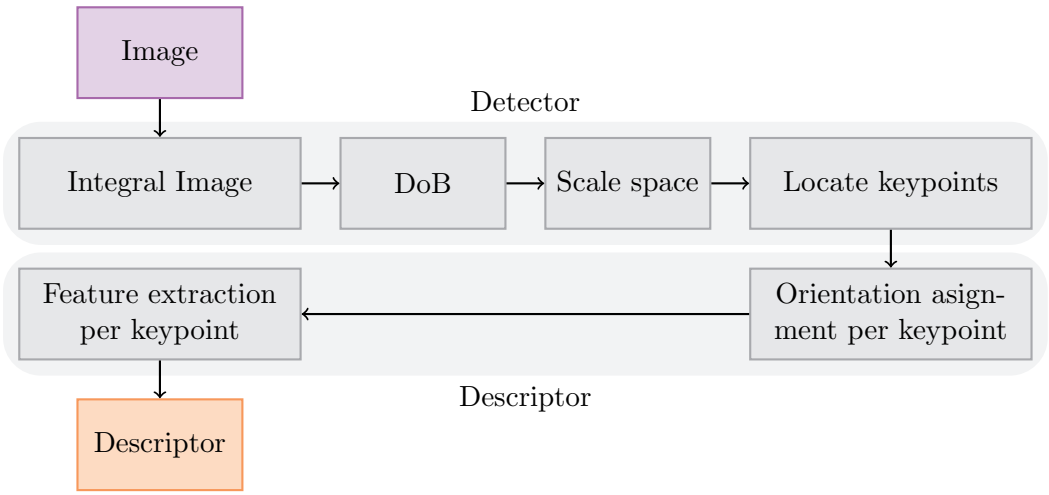


Figura 3.16: Diagrama con las etapas seguidas para obtener los descriptores SURF de una imagen. El proceso se divide en dos fases: detección y descripción.

- En SIFT, en lugar de calcular el *Laplacian of Gaussian (LoG)*, se realiza una aproximación denominada *Difference of Gaussians*. En SURF se realiza una aproximación más eficiente denominada *Difference of Boxes (DoB)*, que nosotros denominaremos diferencia de bloques. La mejoría en la eficiencia se debe al uso de las imágenes integrales para el cálculo de la suma de intensidades de los bloques. Esta técnica fue introducida por Viola y Jones [43].
- En SIFT se realizan iterativamente convoluciones de la imagen de entrada con una máscara Gaussiana $G(x, y, k\sigma)$ para obtener la imagen suavizada a diferentes escalas. Debido a la Diferencia de Bloques *(DoB)*, en SURF no es posible obtener las distintas escalas variando directamente el tamaño de los bloques. De este modo se crea un espacio escalado de forma piramidal (de modo similar a SIFT).

A continuación se explica el propósito de cada uno de los bloques del diagrama de la Figura 3.16, así como sus detalles de implementación.

A) Imagen integral

Este es el cambio principal introducido en SURF, y su uso en fases posteriores consigue una disminución notable del tiempo de computación. El valor de la imagen integral $I_\Sigma(\mathbf{x})$ para el punto $\mathbf{x} = (x, y)$ representa la suma de todos los valores de la región rectangular definida por el origen y el punto $\mathbf{x}$.

$$I_{\Sigma}(x,y) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(i,j) \tag{3.26}$$

Calculada la imagen integral $I_{\Sigma}(\mathbf{x})$, el tiempo para el cálculo de la suma de intensidades de un área rectangular es independiente del tamaño de la imagen, ya que como se explica en la Figura 3.17 sólo depende de tres operaciones. Esta propiedad es muy importante, ya que se usarán filtros de gran tamaño para simular el espacio escalado.

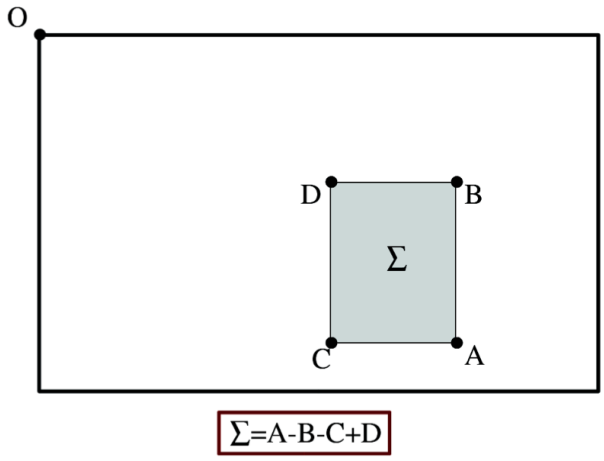


Figura 3.17: Imagen integral. Ejemplo del uso de imágenes integrales para el cálculo de la suma de intensidades en áres rectangulares mediante sumas y restas. Imagen tomada de [39].

B) Diferencia de Bloques *(DoB)*

Para la detección de los puntos de interés se usa una matriz Hessiana. Dado un punto $\mathbf{x} = (x, y)$ en una imagen $I(\mathbf{x})$ y una escala σ, la matriz *Hessiana* $\mathbf{H}(\mathbf{x}, \sigma)$ viene definida por la ecuación 3.27. $L_{xx}(\mathbf{x}, \sigma)$ es la convolución de la derivada de segundo orden de la máscara Gaussiana con la imagen $I(\mathbf{x})$ en el punto $\mathbf{x} = (x, y)$. De forma similar se definen los términos para $L_{xy}(\mathbf{x}, \sigma)$ y $L_{yy}(\mathbf{x}, \sigma)$.

$$\mathbf{H} = \begin{bmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) \\ L_{yx}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma) \end{bmatrix} \tag{3.27}$$

Los elementos anteriores son los mismos que los aproximados en SIFT por David Lowe [38], pero en SURF se decidió hacer una aproximación aún más sencilla basada en filtros definidos por bloques rectangulares (*DoB*). En la Figura 3.18 se muestra un ejemplo de esta aproximación. El uso de este tipo de filtros, en combinación con la imagen integral $I_{\Sigma}(\mathbf{x})$, permite calcular de forma muy eficiente la suma de intensidades en áreas rectangulares (bloques).

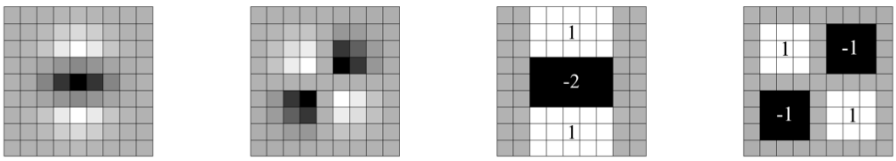


Figura 3.18: Aproximación del *LoG* mediante el uso de *DoB*. De izquierda a derecha: derivadas de segundo orden de la Gaussiana en las direcciones L_{yy} y L_{xy} respectivamente. Aproximación mediante el uso de *DoB* de las derivadas de segundo orden de la gaussiana D_{yy} y D_{xy} respectivamente. El valor de las regiones grises es cero.

C) Espacio escalado

En SIFT, el espacio escalado se crea suavizando la imagen de forma iterativa con máscaras Gaussianas y diezmándolas para crear nuevos niveles de la *space-scale pyramid*. Debido al uso de la imagen integral $I_{\Sigma}(\mathbf{x})$ y máscaras definidas por bloques rectangulares, podemos aplicar máscaras cuadradas de cualquier tamaño directamente sobre la imagen original $I(\mathbf{x})$ sin aumentar el coste computacional de los cálculos. Por ello, el espacio escalado es analizado aumentando el tamaño del filtro en lugar de reduciendo la imagen de forma iterativa como hacíamos en SIFT. Un ejemplo de esto se puede observar en la Figura 3.19.

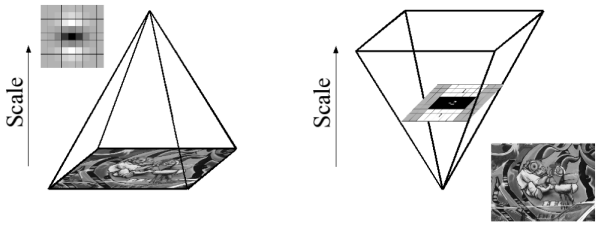


Figura 3.19: Espacio escalado SIFT vs. SURF. Creación del espacio escalado en SIFT mediante la reducción de forma iterativa del tamaño de la imagen (izquierda) y creación del espacio escalado en SURF mediante el uso de la imagen integral aumentando el tamaño de la máscara (derecha). Imagen tomada de [39].

Los resultados son agrupados por octavas. Las octavas se solapan para cubrir de forma total toda la escala. Herbert Bay en su implementación de SURF [39] usa un total de 3 octavas con posibilidad de extenderlas a cuatro y el número de píxeles que engloba cada máscara *DoB* viene determinado por la Figura 3.20. Nótese que las máscaras tienen un número impar para garantizar la presencia de un píxel central.

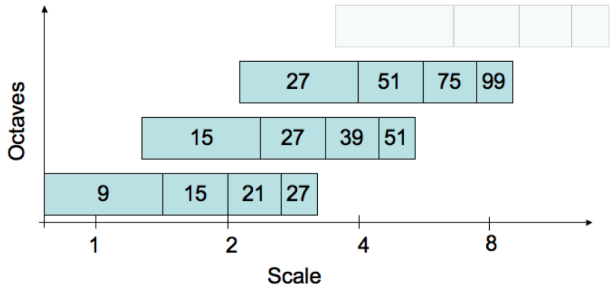


Figura 3.20: Octavas en SURF. En esta representación observamos las 3 octavas usadas en SURF (eje vertical) y las distintas escalas usadas (eje horizontal). Los números representan el total de píxeles de los bloques *DoB* usados. Cómo ejemplo orientativo el valor 9 equivale a un máscara de 3x3. Imagen tomada de [39]

D) Localización de puntos de interés

El procedimiento es similar al ya presentado para SIFT, pero en este caso la serie de expansión de Taylor se aplica a la matrix Hessiana definida anteriormente, obteniendo la Ecuación 3.28.

$$\mathbf{H}(\mathbf{x}) = \mathbf{H} + \frac{\partial \mathbf{H}^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 \mathbf{H}}{\partial^2 \mathbf{x}}\mathbf{x} \tag{3.28}$$

La localización final, más precisa, del extremo $\hat{\mathbf{x}}$ se halla derivando la expansión de Taylor respecto de $\mathbf{x} = (x, y, \sigma)$ e igualándola a 0. Su expresión viene definida por la Ecuación 3.29.

$$\hat{\mathbf{x}} = -\frac{\partial^2 \mathbf{H}^{-1}}{\partial^2 \mathbf{x}}\frac{\partial \mathbf{H}}{\partial \mathbf{x}} \tag{3.29}$$

Cuando nos encontramos con un extremo en las primeras octavas, el área que cubre el bloque rectangular es muy grande. El tamaño de dicho bloque introduce un error significativo en la posición real del punto de interés. Para remediar esto la localización exacta del punto de interés es interpolada ajustando una curva cuadrática 3D en el espacio escalado.

E) Asignación de la orientación

Con el fin de ser invariante frente a rotaciones, el siguiente paso es identificar la orientación de los puntos de interés previamente obtenidos. Para ello se selecciona una región de radio 6σ, es decir, 6 veces la escala a la que fue detectado el punto de interés. A continuación se construye la distribución de las *Haar Wavelets* en las direcciones x e y, aprovechando la eficiencia del uso de la imagen integral y el resultado se pondera con una máscara Gaussiana centrada en el punto de interés y de anchura 2σ .

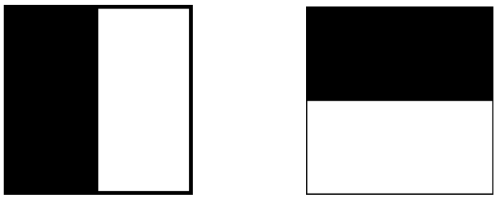


Figura 3.21: Haar Wavelets, para las direcciones x e y.

Los resultados de aplicar las *Haar Wavelets* en las direcciones x e y son representados en un eje cartesiano. La orientación dominante es estimada calculando la suma de las respuestas en una ventana de ángulo $\frac{\pi}{3}$ radianes, como muestra la Figura 3.22. Tras un barrido para todas las posibles ventanas, la dirección con mayor magnitud se añade como nuevo parámetro al punto de interés.

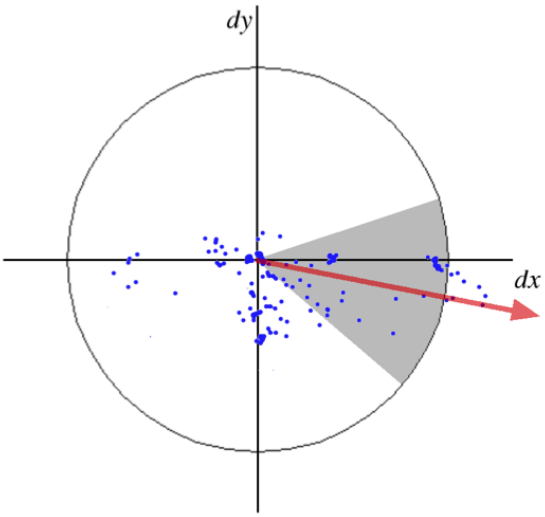


Figura 3.22: Asignación de orientación del punto de interés. Los puntos azules representan las respuestas al aplicar las *Haar Wavelets* en ambas direcciones. La dirección dominante se determina sumando las respuestas contenidas en la ventana de ángulo $\frac{\pi}{3}$ y escogiendo la de mayor magnitud. Imagen tomada de [39].

F) Cálculo de descriptores

Para extraer los parámetros que definen la región alrededor del punto de interés se selecciona una región cuadrada de tamaño 20σ. Esta región es dividida en 4x4 subregiones, por lo que cada subregión tiene un tamaño de 5σ. A continuación se calculan las *Haar Wavelets* para 25 puntos equiespaciados en cada subregión. Denominaremos d_x a la respuesta en dirección horizontal y d_y a la respuesta en la dirección vertical, como se indica en la Figura 3.23.

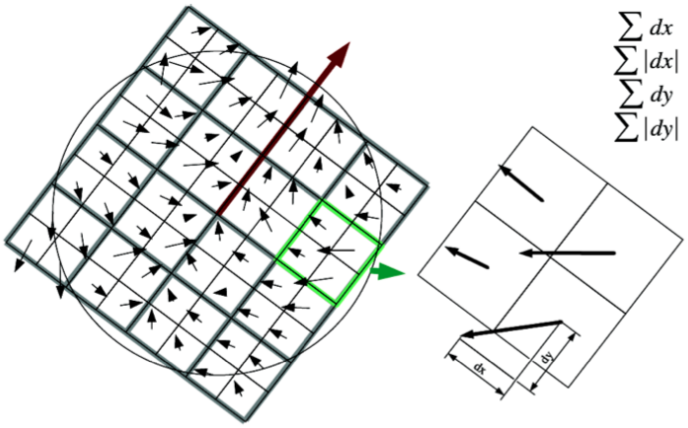


Figura 3.23: Cálculo del descriptor SURF. La región seleccionada, cuyo tamaño es 20σ, se divide en subregiones de 4x4 píxeles. Cada subregión se divide en 5x5 puntos equiespaciados y se aplican las *Haar Wavalets* (izquierda). Las divisiones 2x2 corresponden con los parámetros $\sum dx$, $\sum |dx|$, $\sum dy$ y $\sum |dy|$ calculados para dicho descriptor. Imagen tomada de [39].

Para cada una de las 16 subregiones se calcula un vector descriptivo de 4 parámetros definido por $\mathbf{v} = (\sum dx, \sum dy, \sum |dx|, \sum |dy|)$. La Figura 3.24 ofrece una interpretación de lo que representan estos parámetros. Para una imagen de valor constante, las variaciones dx y dy serán nulas. Si la imagen tiene líneas verticales blancas y negras alternas, tendremos mucha variación en la dirección x. No obstante, los gradientes negro-blanco y blanco-negro ofrecen valores opuestos y se anularán al calcular $\sum dx$, pero no sucederá lo mismo al considerar los valores absolutos, por lo que el valor de $\sum |dx|$ será muy alto. Finalmente, concatenando los vectores $\mathbf{v}$ (compuestos por 4 elementos) para cada una de las 4x4 subregiones, obtenemos un *vector descriptor* de 64 elementos. Las respuestas obtenidas al aplicar las *Haar Wavelets* son invariantes a cambios en iluminación. Para conseguir invarianza frente a cambios de contraste se normaliza el vector obtenido.

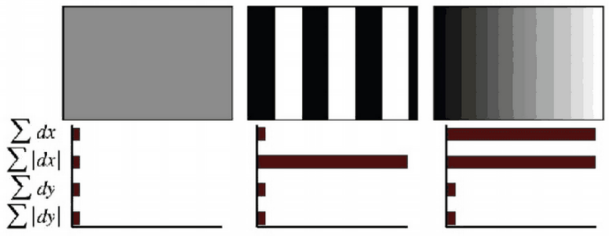


Figura 3.24: Parámetros del descriptor SURF. Ejemplo de los valores del descriptor para diferentes tipos de subregiones. Imagen tomada de [39].

3.2.4. Emparejamiento

Tras obtener los *vectores descriptores* asociados a los puntos de interés y recogidos en el denominado *descriptor* de la imagen, es necesario buscar las correspondencias entre dos imágenes. Dadas dos imágenes I_1 e I_2, tenemos que emparejar un punto de interés de la imagen I_1 con un punto de interés de la imagen I_2 mediante la comparación de sus vectores descriptores. Hay varios métodos para buscar estas correspondencias; a continuación presentamos algunos de ellos.

A) K-NN (K-Nearest Neighbors)

Es un método de clasificación supervisada que parte de un conjunto de datos previamente clasificado a partir del cual se estima la probabilidad a posteriori de que un nuevo elemento e_n pertenezca a cada una de las clases. Este esquema no hace ninguna suposición sobre la distribución inicial de las variables a predecir, asignando el elemento e_n a la clase C_i para la que hay más ejemplos entre los K vecinos más próximos (votación por mayoría).

B) FLANN

El esquema K-NN es determinista: fijado el conjunto de datos, un elemento e_{sample} siempre será asignado a la misma clase C_{sample}. En ciertas aplicaciones no es necesario obtener como respuesta a un nuevo elemento e_{sample} la misma clase C_{sample}, siendo suficiente con obtenerlo en un porcentaje alto de las veces. Esta aproximación permite mejorar notablemente el uso de memoria y el tiempo de computación del algoritmo.

Las técnicas usadas para esta aproximación organizan los datos de alta dimensionalidad usando determinadas estructuras complejas para que la posterior búsqueda sea eficiente. Un ejemplo de estructura sería un árbol, de modo que se podría buscar sólo en las ramas más prometedoras para mejorar el tiempo de computación. La estructura usada por Marius Muja y David Lowe [44] en *Fast Learning Aproximated Nearest Neighbors (FLANN)* es un árbol jerárquico construido con el algoritmo de agrupamiento K-medias. La idea básica consiste en separar los datos aplicando K-medias, recursivamente. El conjunto de datos se divide en K *clusters*, y cada uno de éstos en otros K *clusters* repetídamente, parando la recursión cuando el número de elementos en un cluster sea menor que K.

Una vez creada la estructura, ésta se explora usando el metodo *best-bin-first*, es decir, se comienza con la rama más prometedora. Si el resultado obtenido en esa rama cumple ciertos criterios de calidad, definidos por valores umbrales, porcentajes de datos analizados u otros, la rama se devuelve como resultado; en caso contrario se explora la siguiente rama más prometedora.

3.2.5. Classificación y reconocimiento

Tras obtener los emparejamientos entre dos imágenes I_1 e I_2, es posible usar la información proporcionada por estos emparejamientos para realizar diferentes tareas. Explicaremos algunas de ellas a continuación.

A) Clasificación

Un problema que podemos afrontar es la clasificación de objetos. Dicha clasificación puede referirse a dos tipos distintos de objetos, por ejemplo zapatos y gafas, o lo que es más relevante, puede referirse a dos subclases pertenecientes al mismo tipo de objeto. A modo de ejemplo, si se consideran zapatos, se puede distinguir entre sandalias, zapatillas de deporte, zapatos de tacón o mocasines.

Un posible candidato para clasificación de objetos es el enfoque basado en árboles desarollado por Nister y Stewenius [45] y evaluado por Tomasik, Thiha y Turnbull [46]. La idea consiste en agrupar las características extraídas previamente en un vocabulario de palabras visuales llamado *bag-of-words*, que es usado para construir un árbol que representa la relación entre ellas. Por ejemplo, la relación entre nodos padre-hijo indicará que esas palabras visuales están fuertemente relacionadas, mientras que la relación entre nodos situados al mismo nivel tendrá una relación débil. Este árbol puede ser construído usando el esquema K-medias.

La clasificación de nuevas imágenes puede ser realizada usando K-NN o una versión modificada del mismo donde cada voto se pondere según su cercanía a la imagen de referencia.

Este paso no es obligatorio para conseguir reconocimiento de objetos y podría ignorarse. No obstante, entender cómo afrontar este problema puede ayudar no sólo a clasificar objetos en sus distintas subclases, sino a mejorar el proceso de reconocimiento.

B) Reconocimiento

Si las etapas anteriores (detección, descripción y emparejamiento) son efectuadas satisfactoriamente, debería ser factible el reconocimiento de objetos con una sóla vista (portadas de libros, cuadros, logotipos, etc). No obstante, un esfuerzo por particularizar los métodos descritos anteriormente para diversas áreas puede mejorar notablemente el rendimiento y los resultados.

Hasta ahora estamos considerando objetos con una vista canónica o principal, como es el caso de libros y logotipos cuya superficie además es plana. El problema de reconocimiento se complica al considerar objetos que pueden tener distintas vistas de igual importancia. Para afrontar este problema hay

diferentes alternativas, algunas de las cuales pueden tener en cuenta las siguientes ideas: añadir a la base de datos varias vistas para un mismo objeto, proyectar la imagen de entrada para que sea igual que la vista canónica de dicho objeto en la base de datos, o extraer las características teniendo en cuenta información 3D del objeto.

3.3. Cuestiones de interés

Hay muchas preguntas que pueden surgir y a las que dar respuesta en este área. Algunas posibles preguntas a las que intentar dar respuesta con la realización de este proyecto son enunciadas a continuación:

- ¿Qué grado de transformación geométrica puede soportar el algoritmo?
- ¿Son estos métodos válidos para aplicaciones en tiempo real?
- ¿Es este enfoque válido para bases de datos con muchas imágenes?
- ¿Es este enfoque válido para diferentes vistas de un mismo objeto?
- ¿Cómo manejar la diferencia de resolución que puede haber entre imágenes de internet e imágenes captadas con la cámara del móvil?
- ¿Qué aplicaciones podrían ser desarrolladas?

Capítulo 4

Resultados y análisis

Para obtener buenas prestaciones, tanto en tiempo de computación como en calidad de reconocimiento, en este capítulo estudiaremos cómo se comportan los distintos *detectores* y *descriptores*. De este modo podremos seleccionar aquél que, en términos globales, proporcione mejores resultados o, dependiendo de la aplicación, aquéllos que cumplan determinadas propiedades.

En la Sección 4.1 presentaremos algunos resultados preliminares para mostrar los procesos de detección y emparejamiento, así como el efecto que sobre ellos tienen algunas transformaciones básicas. En la Sección 4.2 se presentan una serie de experimentos para evaluar las prestaciones de distintos *detectores* y *descriptores* ante posibles transformaciones de la imagen y así poder seleccionar aquéllos que mejor se ajusten a nuestras necesidades. Comenzaremos en la Subsección 4.2.1 con una comparación del número total de puntos de interés hallados. Analizaremos el número y porcentaje total de emparejamientos correctos en la Subsección 4.2.2. Seguiremos con una evaluación del tiempo de computación empleado en cada etapa de reconocimiento de objetos (detección, descripción y emparejamiento) en la Subsección 4.2.3.

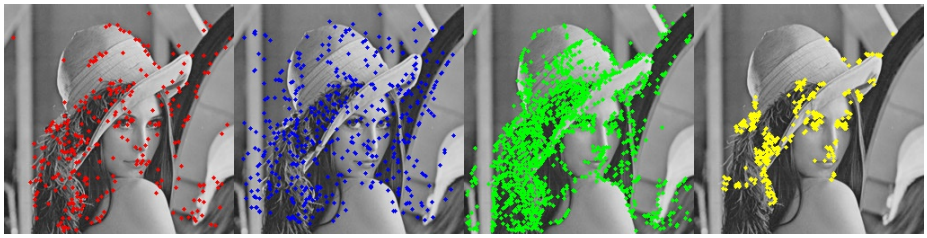
En la Sección 4.3 analizaremos las prestaciones de los algoritmos utilizados para el reconocimiento de objetos sobre un conjunto de imágenes previamente establecido. Trabajaremos con diferentes bases de datos, propias y externas, variando las características de las imágenes y el tamaño de la base de datos. De este análisis se determinará si los algoritmos son suficientemente distintivos para el reconocimiento de objetos en imágenes y si son aptos para clasificación, permitiendo obtener emparejamientos de calidad para imágenes con elementos similares.

4.1. Experimentos preliminares

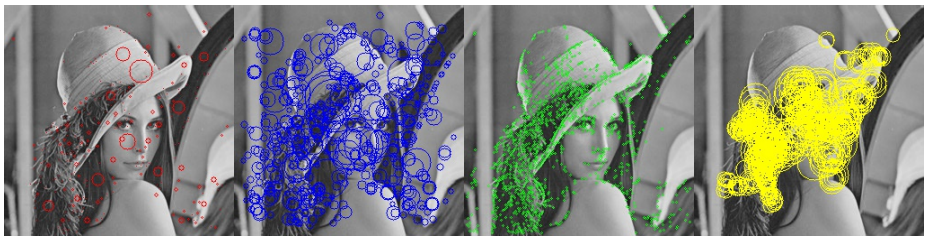
En esta sección mostraremos los resultados parciales obtenidos durante el desarrollo del proyecto, es decir, los puntos de interés, los emparejamientos y cómo éstos se ven afectados ante trasnformaciones de la imagen. Aunque los resultados aquí mostrados no dan información detallada de las prestaciones de los algoritmos estudiados, servirán al lector para entender mejor las gráficas que explicaremos en las secciones posteriores.

4.1.1. Extracción y visualización de los puntos de interés

Los *detectores* se encargan de seleccionar las localizaciones de la imagen que consideraremos de interés, es decir, aquellos puntos que servirán para caracterizar y diferenciar objetos. En la Figura 4.1a se muestran los puntos de interés detectados para la imagen *Lena* de tamaño 256×256, y en la Figura 4.1b se representa con un círculo (potencialmente de distinto tamaño para cada punto de interés) la región de interés asociada a cada punto. Esta región está directamente relacionada con la escala, perteneciente al *scale-space pyramid*, en la que el punto de interés fue detectado. Los algoritmos usados son SIFT [38], SURF [39], FAST [31] y ORB [35], respectívamente.



(a) Localización de los puntos de interés



(b) Región de interés asociada a los puntos

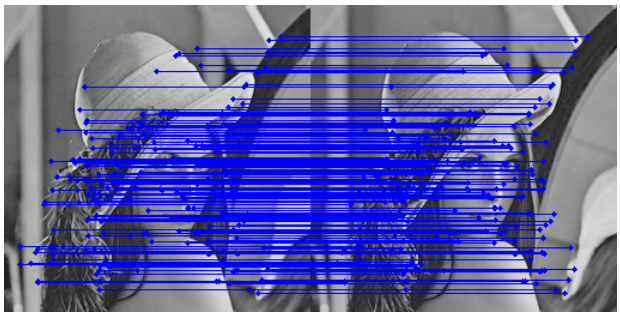
Figura 4.1: Puntos de interés obtenidos para la imagen *Lena* con distintos detectores (a) y región asociada (b). De izquierda a derecha, los algoritmos usados son SURF, SIFT, FAST y ORB.

De la Figura 4.1 se puede observar que el detector FAST determina más puntos de interés, pero en contraposición, la región de interés de los mismos es constante. El detector ORB, sin embargo, es el que detecta menos puntos de

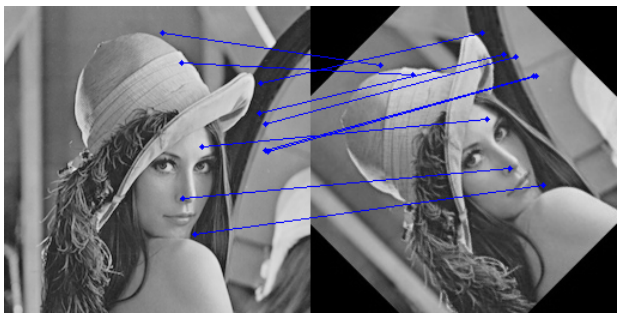
interés, concentrados en zonas que *a priori* podrían considerarse relevantes (tales como ojos, nariz o boca). A medio camino entre FAST y ORB se encuentran los algoritmos SIFT y SURF previamente estudiados.

4.1.2. Ejemplo básico de emparejamiento

A continuación se muestra de forma gráfica el concepto de emparejamiento, que permite asociar un punto de interés de la imagen de entrada con un punto de interés de la imagen de referencia (en principio, almacenada en una base de datos). En la Figura 4.2 se muestra la imagen de referencia (izquierda), y la imagen de entrada (derecha). En la Figura 4.2a se muestran los 200 mejores emparejamientos SURF, considerando distancias euclídeas entre los *vector descriptors* que describen los puntos de interés. Puesto que en la mayoría de casos reales la imagen de entrada y la imagen de referencia serán diferentes, la Figura 4.2b muestra los 10 mejores emparejamientos cuando una de las imágenes es una versión rotada de la otra.



(a) Ejemplo básico de emparejamiento



(b) Ejemplo básico de emparejamiento con rotación

Figura 4.2: Ejemplos de emparejamiento SURF: (a) 200 mejores emparejamientos cuando la imagen de entrada coincide con la de referencia; (b) 10 mejores emparejamientos entre *Lena* y una versión rotada de la misma.

Las transformaciones entre las imágenes a comparar dificultarán el proceso de reconocimiento y serán evaluadas en la Sección 4.2.

4.1.3. Efecto de las transformaciones en la imagen de entrada

En esta sección visualizaremos cómo afectan diferentes transformaciones (escalado, iluminación, difuminado y rotación) de la imagen de entrada (en nuestro caso *Lena*) a los procesos de detección y emparejamiento. Posteriormente, en la Subsección 4.2.2 analizaremos estos efectos en detalle.

A) Efecto del escalado

En este apartado explicaremos cómo afecta el escalado de una imagen a sus emparejamientos. En la Figura 4.3 hemos representado los emparejamientos SURF (fila superior) y SIFT (fila inferior) entre la imagen *Lena* de tamaño 256×256 y varias versiones escaladas de la misma, con factores de escalado 1, 2, 3 y 4. Para una imagen de tamaño $R \times C$, el factor de escalado s modifica su tamaño a $(R/s) \times (C/s)$ píxeles.

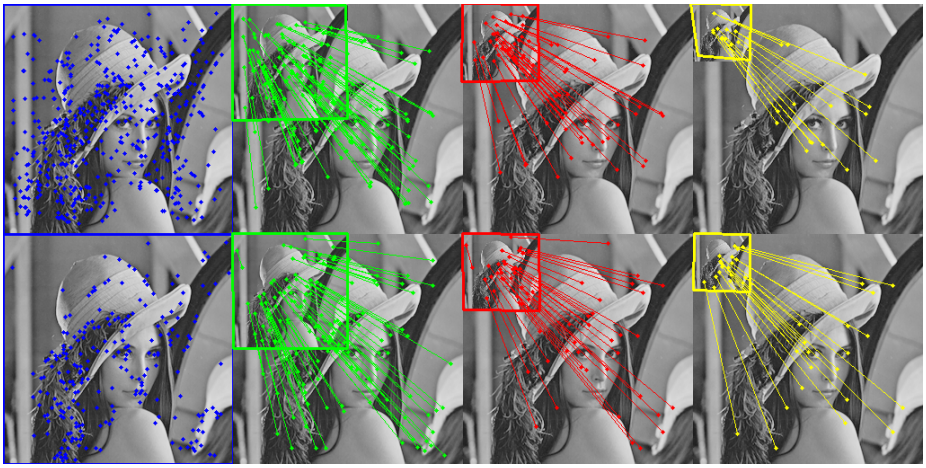


Figura 4.3: Emparejamiento de puntos de interés SURF (fila superior) y SIFT (fila inferior) para diferentes escalas de la imagen *Lena*. Los factores de escalado representados de izquierda a derecha son 1, 2, 3 y 4. El recuadro de color alrededor de la imagen escalada representa la homografía.

La imagen escalada se muestra en la esquina superior izquierda de cada imagen de la Figura 4.3, recuadrada por su homografía. Se observa que el número de emparejamientos disminuye al aumentar el factor de escalado. Además, el descriptor SIFT es más robusto que SURF, obteniendo un mayor número de emparejamientos para el factor de escalado 4 (representado en amarillo) y una homografía más precisa.

B) Efecto de la iluminación

Hay diversos cambios de iluminación que pueden afectar a una imagen. Consideraremos aquí el cambio de iluminación más sencillo, correspondiente a

un *offset* constante que se añade al valor de intensidad de cada píxel. Si definimos los valores de intensidad de los píxeles de una imagen genérica, $I(x, y) = i_{xy}$, la imagen resultado al aplicar un cambio de iluminación será $I'(x, y) = i_{xy}+offset$. El valor se saturará por los extremos, tal como indica la Ecuación 4.1.

$$I'(x,y) = \begin{cases} 0 & \text{si } i_{xy} + \text{offset} < 0 \\ 255 & \text{si } i_{xy} + \text{offset} > 255 \\ i_{xy} + offset & \text{de otro modo} \end{cases} \quad (4.1)$$

En la Figura 4.4 se muestran los emparejamientos SURF para los *offsets* -125, -75, 0 y 50.

Figura 4.4: Emparejamiento de puntos de interés SURF para diferentes transformaciones de iluminación de la imagen *Lena*. Los *offsets* representados de izquierda a derecha son -125, -75, 0, 50. El recuadro alrededor de la imagen representa la homografía.

Para la correcta visualización de la imagen tras aplicar el cambio de iluminación, se han mostrado únicamente 20 emparejamientos. A simple vista no podemos derivar ninguna conclusión, pero intuitivamente el número total de emparejamientos disminuirá al aumentar el $|offset|$.

C) Efecto de difuminado

Para modelar el efecto de difuminado usaremos un filtro espacial Gaussiano de tamaño $S \times S$ (S impar para tener un píxel central) y desviación típica σ. En la Figura 4.5 hemos representado los emparejamientos SURF para $\sigma = 9$ y tamaños S=1, 5, 15 y 41.

Para la correcta visualización de la imagen filtrada se han mostrado únicamente 20 emparejamientos. Se comprueba que el número de emparejamientos

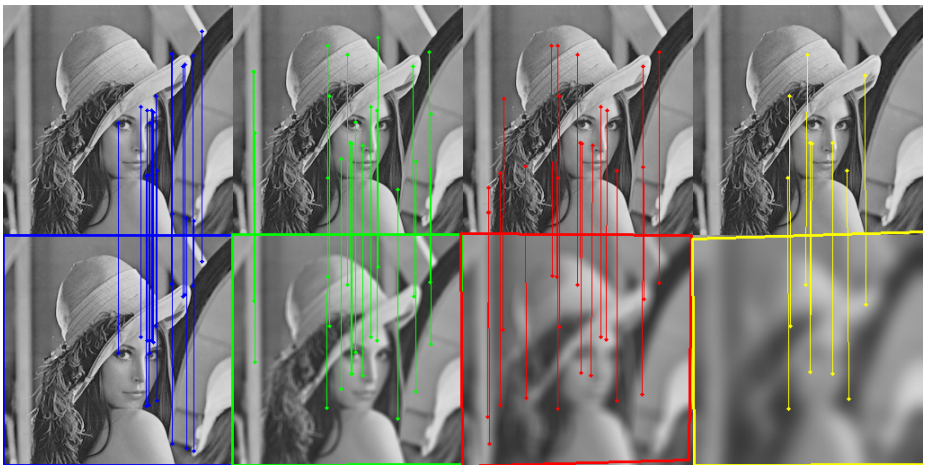


Figura 4.5: Emparejamiento de puntos de interés SURF para diferentes filtrados de *Lena* con máscaras Gaussianas de $\sigma = 9$. De izquierda a derecha, el valor de S que define el tamaño de la máscara es 1, 5, 15 y 41. El recuadro alrededor de la imagen representa la homografía.

disminuye al aumentar el tamaño de la máscara, de modo que en la imagen de la derecha hay menos de 20 emparejamientos.

D) Efecto de la rotación

En este apartado explicaremos cómo afecta la rotación de una imagen a sus emparejamientos. En la Figura 4.6 se muestra, para cada una de las imágenes de *Lena*, los emparejamientos realizados entre la imagen de referencia y la misma imagen rotada un cierto número de grados tomando como punto de rotación el centro de la imagen.

La intuición indica que al rotar la imagen aparecerá en las líneas de emparejamiento un patrón circular. Esto se debe a que los puntos de interés rotarán describiendo una circunferencia, como se puede observar en la primera fila de la Figura 4.6.

Las rotaciones de 90 y 270 grados son casos especiales donde no se ve un patrón circular sino cuadrado: al girar dichos grados, los puntos en el eje de abscisas pasarán a estar en el eje de ordenadas y viceversa, de modo que el trazado de líneas de emparejamiento mostrará un patrón cuadrado. Este comportamiento se puede observar en los emparejamientos realizados en colores cyan y blanco en la Figura 4.6.

En la rotación de 180 grados es de esperar que los puntos de interés entre las dos imágenes se sitúen en posiciones simétricas respecto al punto de rotación, y por lo tanto las líneas de emparejamiento pasarán por el centro.

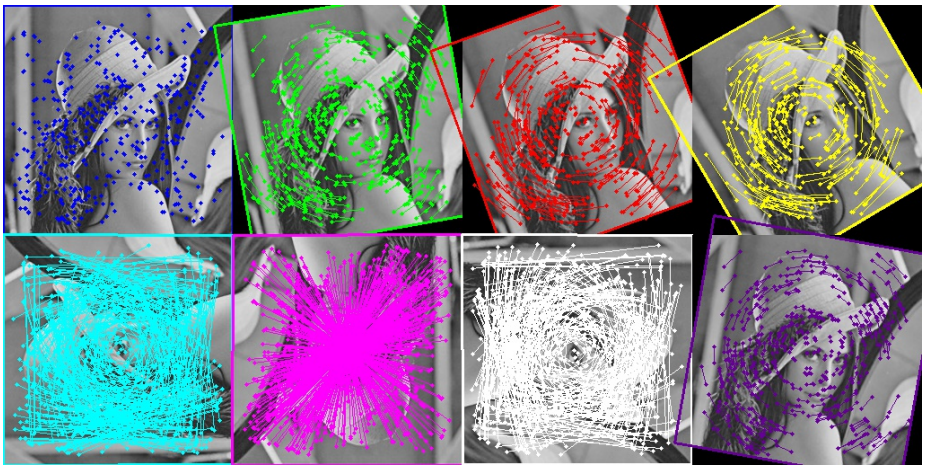


Figura 4.6: Emparejamiento de puntos de interés SURF para diferentes rotaciones de la imagen de *Lena*. Los ángulos de rotación representados, de izquierda a derecha y de arriba a abajo son 0, 10, 20, 30, 90, 180, 270 y 350. El recuadro alrededor de la imagen representa la homografía.

Este comportamiento se puede observar en los emparejamientos realizados en color rosa en la Figura 4.6.

4.2. Análisis de prestaciones

En esta sección se presentan las pruebas realizadas para analizar las prestaciones de los algoritmos de detección y descripción de características visuales. Detallaremos las condiciones en las que se realizaron estos análisis, las conclusiones que se obtuvieron y las decisiones tomadas a partir de ellos.

4.2.1. Detección de puntos de interés

El primer parámetro que podemos analizar es el número de puntos de interés encontrados para cada uno de los *detectores*. En la Figura 4.7 se muestra el número total de puntos de interés con distintos tipos de detectores y para distintos tamaños de la imagen *Lena* (64, 128, 256, 512). Como ya intuíamos, el detector FAST es el que más puntos localiza y nuevamente los descriptores SURF y SIFT se sitúan en un término medio.

La gráfica muestra que con imágenes de tamaño reducido el detector STAR no localiza ningún punto de interés, hecho que imposibilitaría la realización del emparejamiento y por tanto no se obtendría ningún resultado para la tarea de reconocimiento. Otros detectores, como por ejemplo BRISK y ORB, detectan muy pocos puntos cuando el tamaño de la imagen es reducido, lo que también dificulta el proceso de reconocimiento. Por este motivo se descartan los detectores

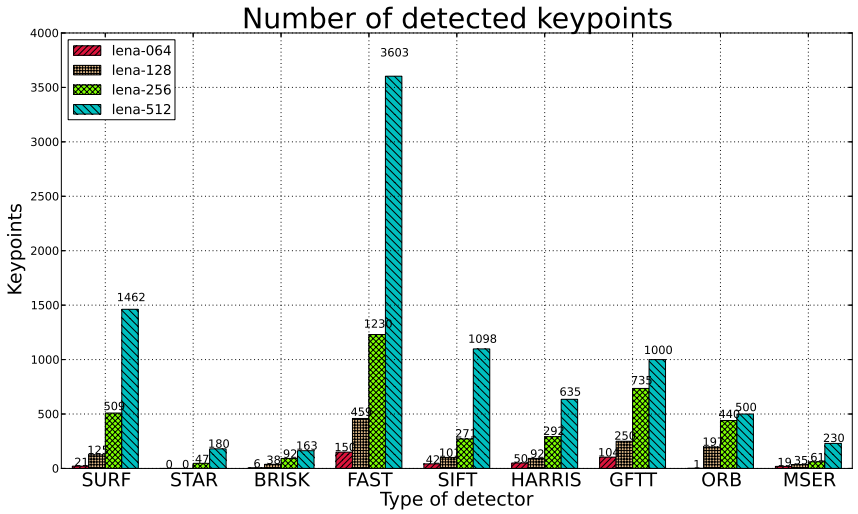


Figura 4.7: Número total de puntos de interés localizados por cada detector. Los tamaños de las imágenes cuadradas usadas han sido 64, 128, 256 y 512.

para los que el número total de puntos de interés sobre la imagen *Lena* de tamaño 64×64 es muy reducido.

Los algoritmos que seguiremos evaluando son SURF, SIFT y FAST. Aclarar que FAST es un algoritmo de detección de puntos de interés, pero no de descripción, por lo que tendremos que combinarlo con la fase de descripción SIFT y/ó SURF. En lo que sigue, la notación que usaremos será *detector-descriptor*.

4.2.2. Robustez frente a transformaciones

Otro factor importante es la robustez de los puntos de interés y de sus *vectores de descripción* ante posibles transformaciones de la imagen de entrada. En las siguientes pruebas hemos transformado la imagen de *Lena* simulando dichas transformaciones y hemos representado dos tipos gráficos. En el primer tipo representaremos el número de emparejamientos correctos para diferentes valores de una misma transformación. En el segundo tipo representaremos el porcentaje de emparejamientos correctos respecto del número total de puntos de interés detectados.

Debido a que conocemos la transformación realizada, también conocemos la denominada *ground truth*, es decir, qué puntos han de ser emparejados entre la imagen original y la imagen transformada. Para determinar si un emparejamiento es correcto hemos delimitado una *región de tolerancia* de 3×3 píxeles alrededor del punto de interés. Definamos como $\mathbf{p} = (x, y)$ el punto de interés en la imagen original, de modo que la localización deseada de $\mathbf{p}$ en la imagen transformada es $\mathbf{p}' = (x', y')$, siendo $\mathbf{t}$ el punto de interés en la imagen transformada que el

algoritmo determina como asociado a $\mathbf{p}$. Nótese que los puntos $\mathbf{p}$ y $\mathbf{p}'$ pueden tener distintas coordenadas (dependiendo de la transformación), como se ilustra en la Figura 4.8 a través de $\mathbf{p}_3$. Si $\mathbf{t}$ yace en la región de tolerancia centrada en $\mathbf{p}'$, el emparejamiento será considerado correcto (indicado en verde en la Figura 4.8); en caso contrario será considerado incorrecto (color rojo en la Figura 4.8). La región de tolerancia considerada en este trabajo tiene tamaño 3×3.

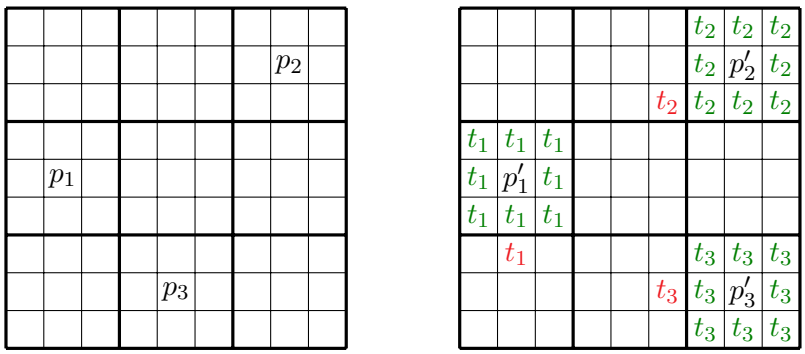


Figura 4.8: Ejemplo de emparejamientos para los 3 puntos de interés $\mathbf{p}_1, \mathbf{p}_2$ y $\mathbf{p}_3$ de la imagen original. Imagen transformada (derecha) con emparejamientos deseados (puntos $\mathbf{p}'$) y emparejamientos reales (puntos $\mathbf{t}$), marcando en verde los potenciales emparejamientos correctos.

En la Subsección 4.2.1 comprobamos que el número total de puntos de interés hallados variaba según el detector considerado. Es por ello que, para realizar una comparación adecuada, analizaremos el porcentaje de emparejamientos correctos respecto al número total de puntos de interés detectados. A continuación se muestran los resultados y conclusiones para distintas transformaciones.

A) Escalado

Es habitual encontrar el mismo objeto a diferente escala en dos imágenes asociadas a la misma escena, bien debido a la distancia desde la que es tomada la imagen o a la configuración del dispositivo que la toma, por ejemplo el *zoom*. En la Figura 4.9 se muestra el número y porcentaje de emparejamientos correctos al escalar la imagen de *Lena*, detectar puntos de interés y realizar emparejamientos con los puntos de interés de la imagen original.

Del análisis de la Figura 4.9 se puede concluir que los algoritmos que usan FAST en la detección de puntos de interés son los que menor porcentaje de emparejamientos correctos proporcionan. El detector-descriptor más resistente al escalado es SURF hasta un factor de escalado de 5, a partir del cual sus prestaciones decaen drásticamente. Por el contrario SIFT muestra un comportamiento más regular con el factor de escalado. Recordemos que la robustez

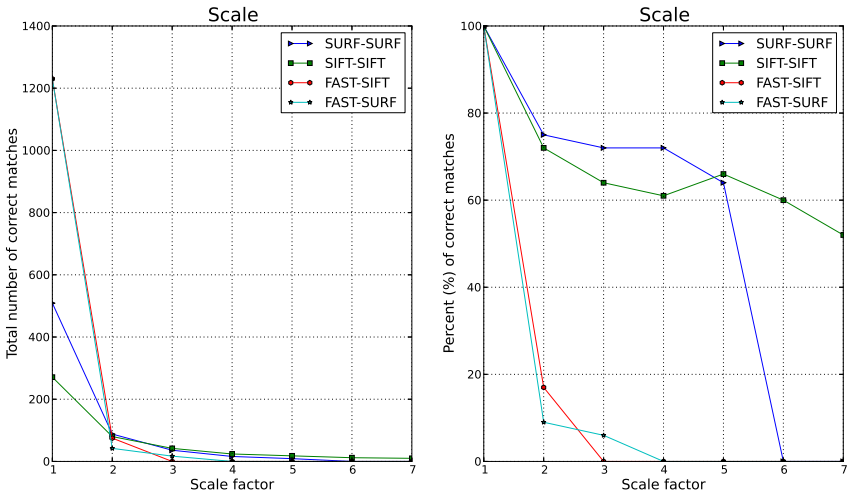


Figura 4.9: Evaluación del porcentaje de emparejamientos correctos (tolerancia 3×3) frente al factor de escalado para cuatro combinaciones *detector-descriptor*.

frente a cambios de escala se consigue mediante la creación del espacio escalado: el espacio escalado definido por Lowe [38] para SIFT usaba 4 octavas, mientras que el definido por Bay [39] para SURF usaba 3 octavas. Por tanto, parece razonable que SIFT ofrezca una respuesta más robusta que SURF ante cambios de escala. Nótese la diferencia en el número total de elementos de los *vectores descriptores*: 128 elementos en SIFT y 64 elementos en SURF. Para imágenes muy difuminadas del espacio escalado, correspondientes a octavas finales y factores de escalado grandes, las regiones asociadas a los puntos de interés de la imagen serán muy parecidas entre sí. Esto es debido a la falta de altas frecuencias en la imagen, siendo el número de elementos del *vector descriptor* relevante para distinguir/discriminar las pequeñas variaciones entre dichas regiones.

B) Iluminación

La siguiente transformación a analizar corresponde a cambios en la iluminación, es decir, a cambios en el nivel de intensidad de los píxeles de la imagen. Los cambios de intensidad pueden aparecer de forma natural por la diferencia entre día-noche o sol-sombra, o puede producirse por elementos artificiales como el *flash* del dispositivo o la iluminación de la escena. En este trabajo únicamente se consideran cambios de iluminación uniformes en toda la escena, obtenidos al añadir un *offset* al nivel de intensidad de todos los píxeles, detectar los puntos de interés de la imagen transformada y emparejarlos con los puntos de la imagen original. El resultado se muestra en la Figura 4.10.

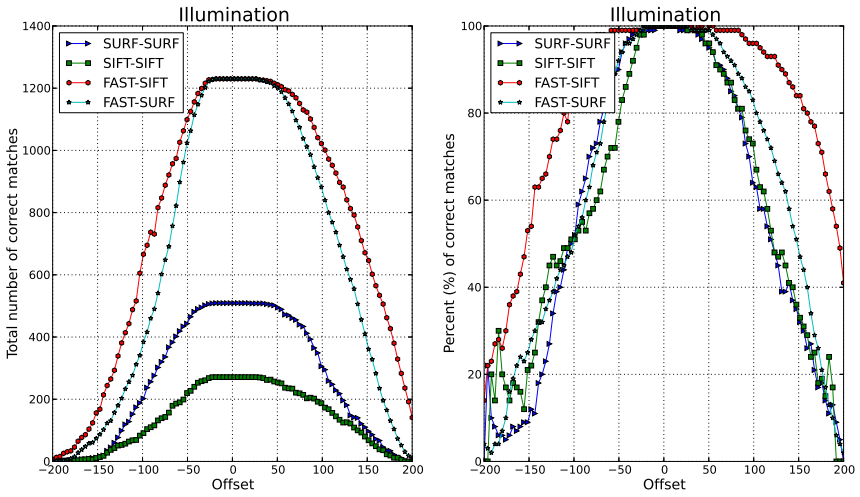


Figura 4.10: Evaluación de los emparejamientos correctos (tolerancia 3×3) frente al factor de *offset* para cuatro combinaciones *detector-descriptor*.

Se comprueba que, para este tipo de transformación, todos los algoritmos ofrecen prestaciones similares, muy buenas cuando el *offset* es bajo y degradándose a medida que el *offset* aumenta (posiblemente debido a una pérdida de contraste de los elementos de la escena) . Recordemos aquí que los esquemas SIFT y SURF normalizaban el vector decriptor para obtener robustez frente a cambios de iluminación.

C) Difuminado

El interés de analizar esta transformación tiene su origen en la adquisición de imágenes con dispositivos de baja calidad. El análisis de este factor, en combinación con el escalado, permite evaluar cómo se comportará nuestro sistema en dispositivos móviles, en especial en aquéllos cuya cámara sea de baja calidad. Para simular el difuminado se utiliza un filtro espacial Gaussiano. Tenemos que tener en cuenta dos parámetros: tamaño de la máscara, que definiremos como $S \times S$, y desviación típica σ de la Gaussiana que estamos aproximando con dicho filtro.

El resultado obtenido al difuminar la imagen, detectar los puntos de interés y emparejarlos con los puntos extraídos de la imagen original se ilustra en la Figura 4.11, donde las combinaciones SURF-SURF, SIFT-SIFT y FAST-SURF se han representado en azul, verde y rojo respectivamente. Puesto que las prestaciones de los algoritmos que usan FAST en la fase de detección no son muy buenas, y para permitir una mejor comparación entre SIFT y SURF, no se ha representado la combinación FAST-SIFT.

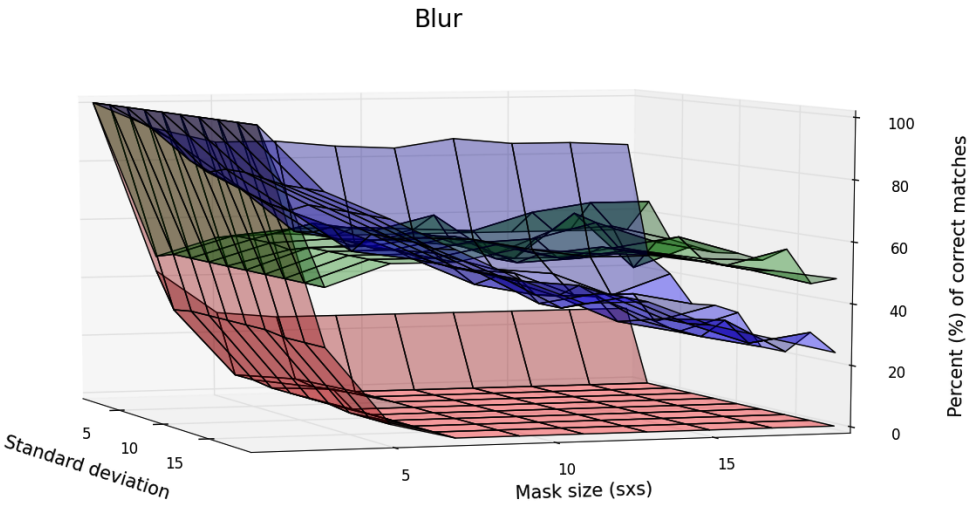


Figura 4.11: Evaluación del porcentaje de emparejamientos correctos (tolerancia 3×3) frente al tamaño ($S \times S$) de la máscara y la desviación típica σ de la Gaussiana para tres combinaciones *detector-descriptor*.

El análisis de los resultados indica que nuevamente los algoritmos que usan FAST presentan un resultado pobre. El mejor resultado es obtenido usando SURF cuando el difuminado es bajo, pero su robustez decae al aumentar el tamaño del filtro espacial y la varianza de la Gaussiana. Por el contrario, el esquema basado en SIFT presenta un comportamiento más estable. Recordemos también que para aproximar el *Laplacian of Gaussian (LoG)*, SIFT usa una aproximación basada en *Difference of Gaussians (DoG)* y SURF usa la aproximación basada en *Difference of Boxes (DoB)*, que es más inexacta que la aproximación de SIFT, lo que da sentido a los resultados obtenidos.

D) Rotación

El análisis de esta transformación tiene su interés en la potencial rotación del objeto o de la cámara al adquirir la imagen. La evolución del porcentaje de emparejamientos correctos obtenidos al rotar la imagen/objeto un determinado número de grados se muestra en la Figura 4.12.

Nuevamente el esquema más estable frente a cambios en la rotación es SIFT. Recordemos que la robustez de los descriptores frente a rotaciones se consigue mediante la asignación de una orientación principal a cada punto de interés, determinada ésta en función del entorno del punto de interés. El esquema SIFT hace uso de un histograma con 36 contenedores, por lo que es posible diferenciar entre 36 orientaciones distintas. Por contra, el esquema SURF caracteriza las orientaciones en ventanas de $\frac{\pi}{3}$ radianes, y por lo tanto

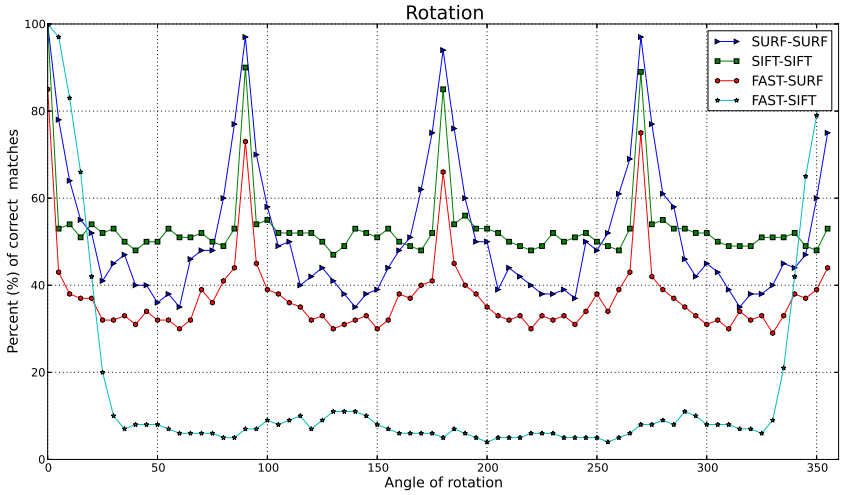


Figura 4.12: Evaluación del porcentaje de emparejamientos correctos (tolerancia 2×2) frente a la rotación (expresada en grados) para cuatro combinaciones *detector-descriptor*.

tendremos 6 ventanas diferentes, es decir, 6 orientaciones principales posibles. Vemos que para 90, 180 y 270 grados obtenemos una tasa de emparejamiento correcto cercana al 100 %.

Tras la realización de los experimentos podemos concluir que los algoritmos más robustos frente a transformaciones entre la imagen de entrada y la de referencia son SURF y SIFT. Si, por el contrario, se tiene certeza que la imagen de entrada es la misma que la imagen de referencia, entonces la mejor opción es usar el detector FAST ya que detecta más puntos de interés en menor tiempo, ofreciendo altos porcentajes de emparejamiento en combinación con otros descriptores. En adelante, en este trabajo se descartó FAST por su poca robustez frente a cambios en la imagen de entrada. Para más información acerca de la robustez de SIFT y SURF consúltese [47] [48].

4.2.3. Tiempo de cómputo

El tiempo de cómputo es un factor clave en aplicaciones de tiempo real. Este tiempo se puede desglosar teniendo en cuenta las tareas a analizar en las tres fases (detección, descripción y emparejamiento) del proceso de reconocimiento de objetos descritas en el Capítulo 3. A continuación se enumeran los tiempos considerados y se proporciona una ligera intuición sobre su importancia.

- Tiempo de detección

  Tiempo necesario para detectar los *puntos de interés* de una imagen. Para un mismo algoritmo, el tiempo de detección dependerá del tamaño de la imagen. Para dos algoritmos distintos el tiempo empleado variará según el método de creación del espacio escalado y las aproximaciones usadas.

- Tiempo de descripción

  Tiempo necesario para calcular el *vector descriptor* con las características de la región asociada a los puntos de interés. El tiempo total de descripción depende del número de puntos de interés hallados y del algoritmo usado.

- Tiempo de emparejamiento

  A tener en cuenta cuando estamos buscando una determinada imagen en una base de datos. En este caso, los puntos de interés y vectores de descripción asociados a las imágenes de la base de datos pueden ser calculados y almacenados previamente. No obstante, siempre será necesario realizar el emparejamiento entre los puntos de interés de la imágen de entrada y los de las imágenes de la base de datos.

Para tener una medida del tiempo de cómputo de los métodos de detección, descripción y emparejamiento se han considerado 10 realizaciones y se ha calculado el tiempo medio. En todos los análisis se ha considerado el caso más sencillo, es decir, imagen de entrada e imagen de referencia son copias exactas, de modo que se realiza el emparejamiento de dos *descriptores* iguales. En las siguientes gráficas se muestran valores de tiempo medio por punto de interés.

La Figura 4.13 muestra una comparativa del tiempo medio de detección por punto de interés para cada uno de los diferentes detectores. Se han omitido los datos correspondientes al detector MSER porque sus altos valores de tiempo de detección dificultaban la comparación visual con el resto de detectores.

De la Figura 4.13 se concluye que FAST es un algoritmo realmente rápido, y que por el contrario SURF y SIFT son lentos en comparación con el resto. Basándonos en el número de puntos de interés detectados (Figura 4.7) y el tiempo de detección (Figura 4.13) hemos tomado la decisión de centrar el estudio en el comportamiento de SIFT y SURF. Las motivaciones principales han sido:

- Bajo número de puntos de interés detectados para la imagen de 64x64 píxeles, debido a que es un requisito indispensable si el tamaño del objeto en la imagen es reducido.

- Relación tiempo de detección y número de puntos de interés. Si el tiempo de detección es muy bajo, es posible que los puntos de interés no sean “de calidad”.

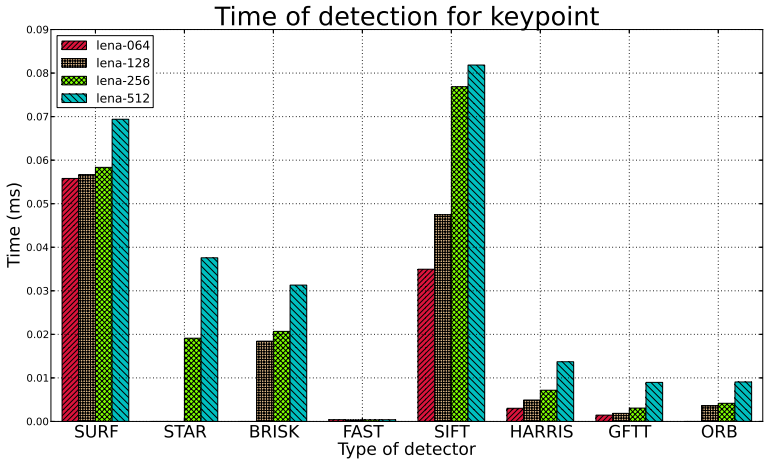


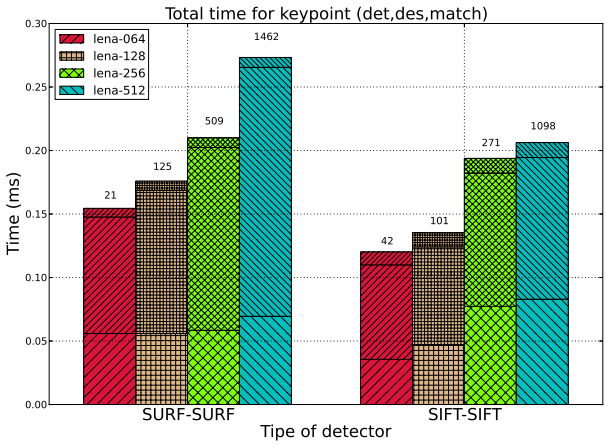
Figura 4.13: Tiempo medio de detección por punto de interés para imágenes cuadradas de distinto tamaño: lado 64, 128, 256 y 512 píxeles.

- El análisis de la robustez presentado en la Subsección 4.2.2 muestra que SIFT y SURF tienen muy buenas prestaciones.

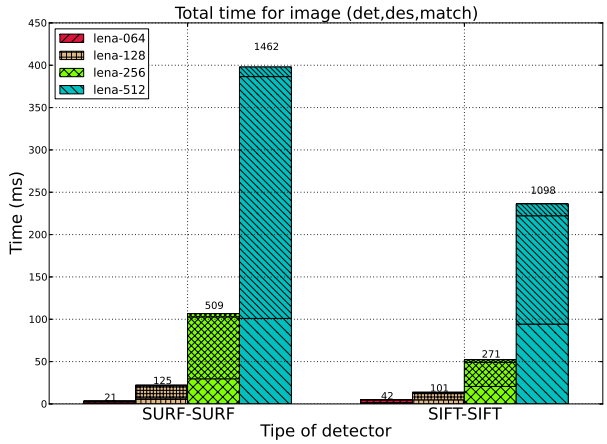
Para una mejor comparación del tiempo de cómputo global, se ha creado un diagrama de barras como los anteriores, pero dividiendo cada contenedor (*bin*) en tres sectores. El sector superior hace referencia al *tiempo de emparejamiento*, el central al *tiempo de descripción*, y el inferior al *tiempo de detección*. En ocasiones el tiempo de detección es tan pequeño que es imposible distinguirlo en el diagrama de barras.

En la Figura 4.14a se representan los tiempos medios de cómputo por punto de interés. Se observa el tiempo total de cómputo y, si nos fijamos en los distintos sectores, el tiempo empleado en cada fase. En la Figura 4.14b se indican los tiempos de cómputo totales, es decir, para todos los puntos de interés extraídos de la imagen.

El propio Bay, en su implementación de SURF [39], destaca que el tiempo de computación de SURF es varias veces más rapido que el de SIFT. No es posible tomar una conclusión contundente basándonos únicamente en la Figura 4.14. Supondremos que la implementación de SURF existente en OpenCV no es eficiente, y por ese motivo el algoritmo SIFT presenta un menor tiempo de computación. Continuaremos evaluando esta asunción para cada una de las bases de datos que hemos creado.



(a) Tiempo medio de cómputo por punto de interés



(b) Tiempo total de computación

Figura 4.14: Tiempo de cómputo para detección, descripción y emparejamiento: (a) tiempo medio por punto de interés; (b) tiempo total considerando todos los puntos de interés extraidos de la imagen. En la parte superior de cada contenedor se indica el número de puntos de interés.

4.3. Evaluación en bases de datos

Una característica importante de los algoritmos de reconocimiento, principalmente para su uso en aplicaciones reales, es su comportamiento cuando se trabaja con bases de datos (en general, de gran tamaño). En esta sección describiremos las bases de datos utilizadas (algunas de ellas de creación propia) y analizaremos el comportamiento de los algoritmos de reconocimiento.

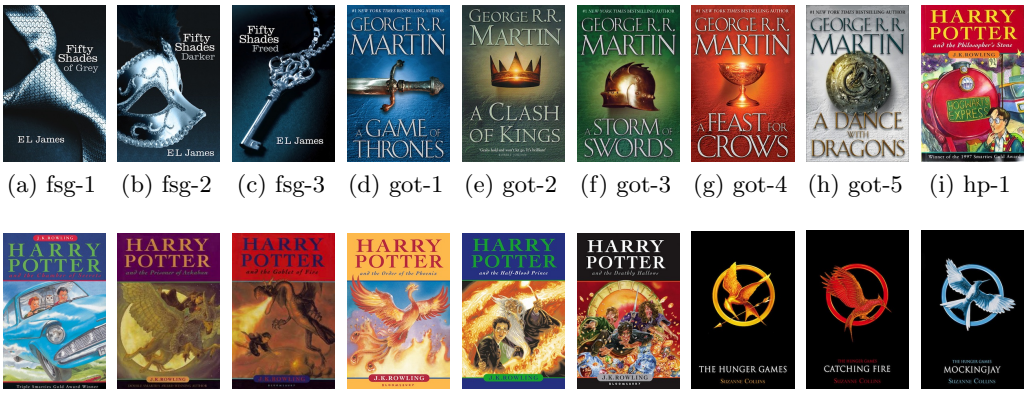
4.3.1. Descripción de las bases de datos

A) BBDD-1:

El objetivo es evaluar las prestaciones de los algoritmos de reconocimiento basados en SIFT y SURF para recuperar imágenes basadas en el contenido de la portada. Con este objetivo se ha construido una reducida base de datos con portadas de libros famosos; en concreto se han elegido cuatro sagas distintas:

- *Fifty shades of Gray* (3 portadas).
- *Game of thrones* (5 portadas).
- *Harry Potter* (7 portadas).
- *The hunger games* (3 portadas).

La base de datos dispone por tanto de 18 portadas, cada una representando la vista canónica del libro. Todos las portadas tienen el mismo tamaño, en este caso 400×262 píxeles, y han sido descargadas de *Google Images*. La base de datos completa se muestra en la Figura 4.15.



(a) fsg-1 (b) fsg-2 (c) fsg-3 (d) got-1 (e) got-2 (f) got-3 (g) got-4 (h) got-5 (i) hp-1

(j) hp-2 (k) hp-3 (l) hp-4 (m) hp-5 (n) hp-6 (ñ) hp-7 (o) thg-1 (p) thg-2 (q) thg-3

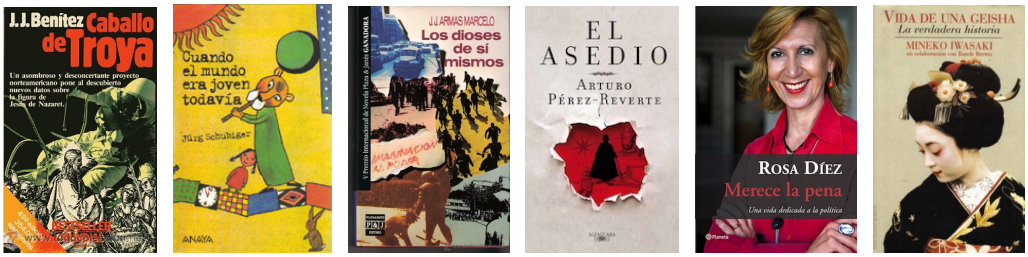
Figura 4.15: Portadas de libros de la base de datos BBDD-1.

B) BBDD-2:

Esta base de datos contiene portadas de libros que tengo físicamente a mi disposición. El objetivo es evaluar las prestaciones de una aplicación de reconocimiento de portadas de libros cuando éstos se capturan a través de una cámara de vídeo en un escenario real. Como es una aplicación de vídeo y no es deseable romper la sensación de "fluidez" del mismo, la base de datos contiene un menor número de libros para reducir el tiempo dedicado al emparejamiento de los puntos de interés. A continuación se indican los libros considerados:

- Caballo de Troya (367×229).
- Cuando el mundo era joven todavía (235×150).
- Los dioses de sí mismos (400×262).
- El asedio (241×150).
- Merece la pena (400×262).
- Vida de una geisha (320×296).

Por lo tanto dispondremos de 6 portadas, cada una representando la vista canónica del libro. Las portadas tienen distintos tamaños, indicados en el listado anterior, y han sido descargadas de *Google Images*. La base de datos completa se muestra en la Figura 4.16.



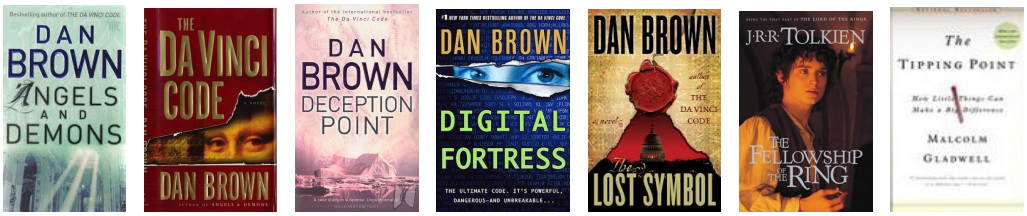
(a) troya (b) mundo (c) dioses (d) asedio (e) pena (f) geisha

Figura 4.16: Portadas de libros de la base de datos BBDD-2.

C) BBDD-3:

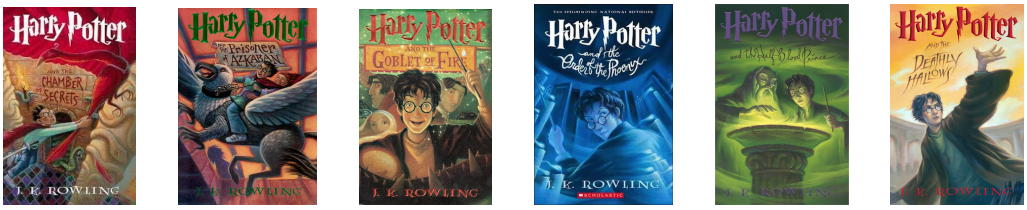
Esta base de datos contiene todas las portadas que aparecen en la imagen de entrada usada para ilustrar el ejemplo de reconocimiento de objetos en imágenes estáticas que se presenta en la Sección 5.2.

La base de datos dispone de 19 portadas, cada una representando la vista canónica de un libro. Las portadas tienen distintos tamaños, desde 143×93 píxeles para la Figura 4.17m hasta 1186×702 píxeles para la Figura 4.17c. Las



(a) DB-1 (b) DB-2 (c) DB-3 (d) DB-4 (e) DB-5 (f) JT-1 (g) MG-1

(h) RK-1 (i) SM-1 (j) SM-2 (k) SM-3 (l) SM-4 (m) SR-1 (n) JK-1



(ñ) JK-2 (o) JK-3 (p) JK-4 (q) JK-5 (r) JK-6 (s) JK-7

Figura 4.17: Portadas de libros de la base de datos BBDD-3.

imágenes han sido descargadas de *Google Images*. La base de datos completa se muestra en la Figura 4.17.

D) BBDD-4:

Esta base de datos contiene también portadas de libros de distintos tamaños, pero en este caso obtenidos de la base de datos OpenLibrary [1]. El objetivo es evaluar las prestaciones de los algoritmos SIFT y SURF en bases de datos con gran número de elementos. Aunque OpenLibrary contiene más de 1.000.000 títulos, para reducir el tiempo de ejecución y mejorar la visualización de los resultados no trabajaremos con todos los títulos. El número de portadas de libros utilizadas será indicado en los propios análisis.

4.3.2. Análisis BBDD-1

Analizaremos el tiempo de computación necesario para cada imagen de la base de datos. De este modo podremos entender mejor cómo varía el tiempo de computación para los esquemas basados en SIFT y SURF. En la Figura 4.18

[1] Link: `http://openlibrary.org`

se muestra el tiempo de detección, descripción y emparejamiento, así como el tiempo total y el número de puntos de interés extraídos para cada imagen de la base de datos.

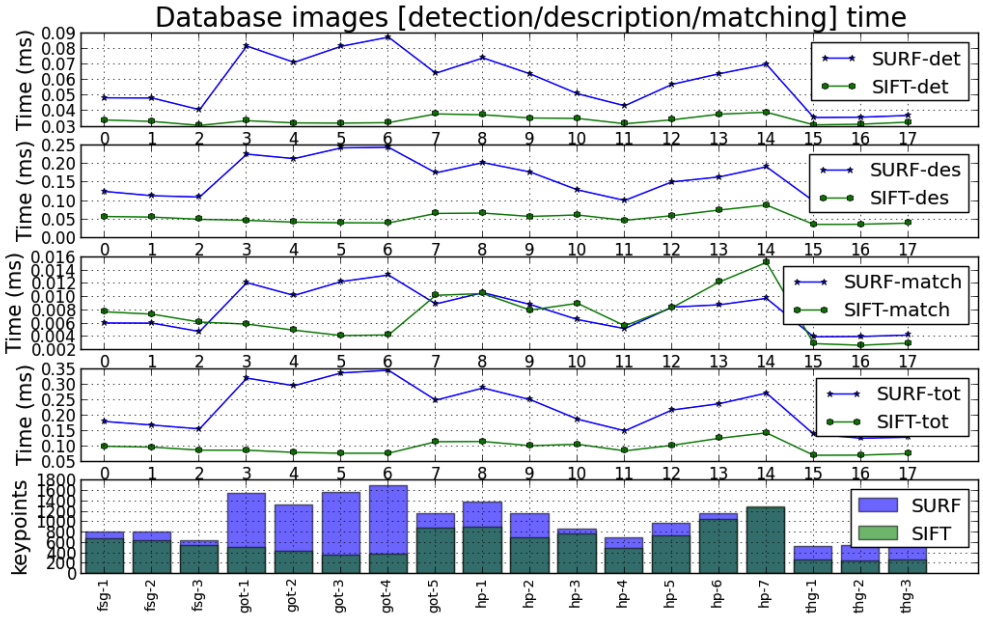


Figura 4.18: Tiempo de detección, descripción, emparejamiento, tiempo total y número de puntos de interés para la BBDD-1 usando SIFT y SURF.

Podemos observar que los tiempos de detección y emparejamiento son despreciables en comparación con el tiempo de descripción. Por este motivo la gráfica que muestra el tiempo total, suma de los tres anteriores, tiene el mismo perfil que la gráfica correspondiente a los tiempos de descripción. El tiempo de cómputo del descriptor SIFT es menor que el de SURF. Nótese también que el tiempo de emparejamiento puede ser un problema si el número de imágenes en la base de datos es muy alto. Para un mismo número de puntos de interés el tiempo de emparejamiento de SIFT es mayor que el de SURF. Recordemos que los *vectores descriptores* de SIFT están formados por 128 elementos, mientras que los de SURF tienen la mitad (64).

Analizando la gráfica inferior de la Figura 4.18 se observa que el número de puntos de interés detectado con cada uno de los algoritmos es claramente diferente en el caso de las portadas de la saga *Game of Thrones*, probablemente debido a la presencia de muchos puntos de interés asociados a las letras (título y autor) que aparecen.

Para evaluar el comportamiento de los algoritmos SIFT y SURF en portadas con elementos comunes, hemos realizado dos pruebas. En la primera hemos definido como imagen de entrada la portada de *Harry Potter III* (fila superior de

la Figura 4.19), y hemos mostrado las 5 portadas de la base de datos con más emparejamientos en común (fila inferior) en la Figura 4.19. Las portadas están ordenadas de izquierda a derecha según número de emparejamientos decreciente. Se observa que el elemento común de la portada es el título de la saga, y todos los emparejamientos hacen referencia al mismo.

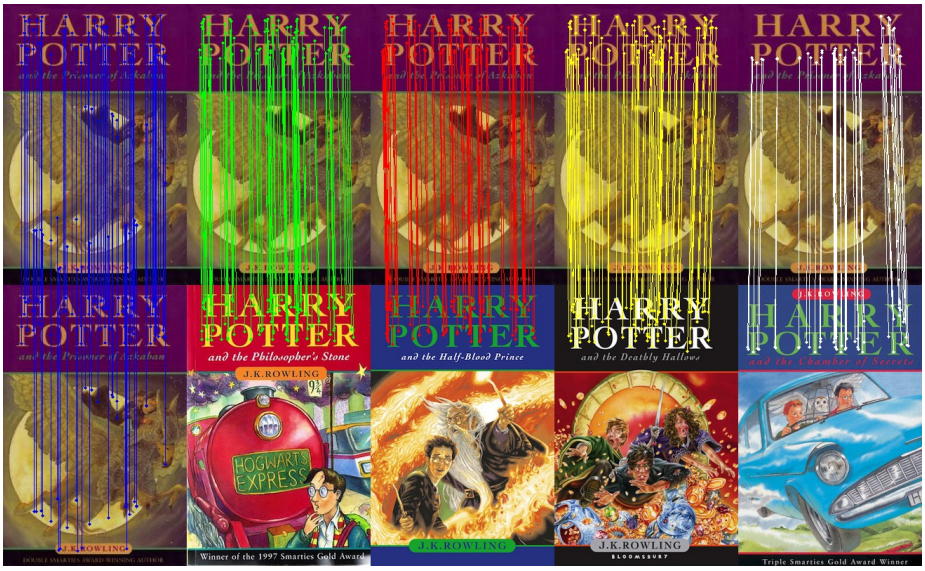
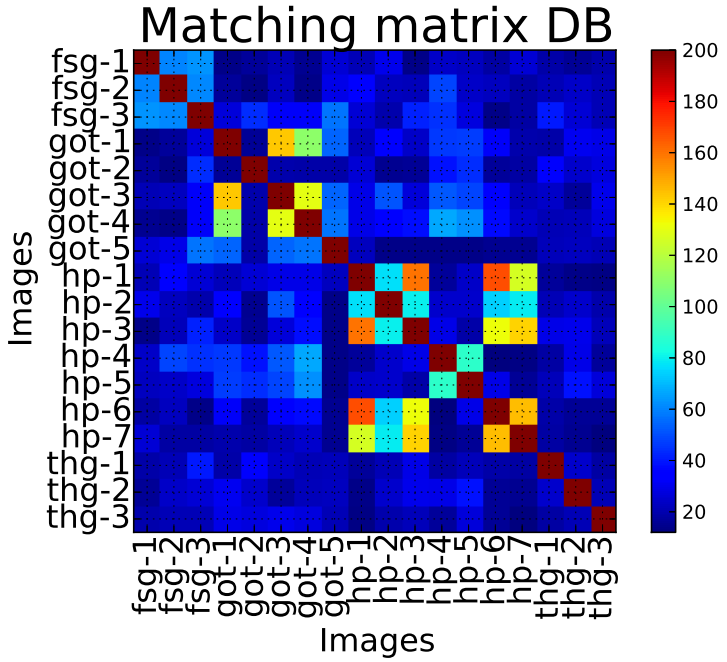


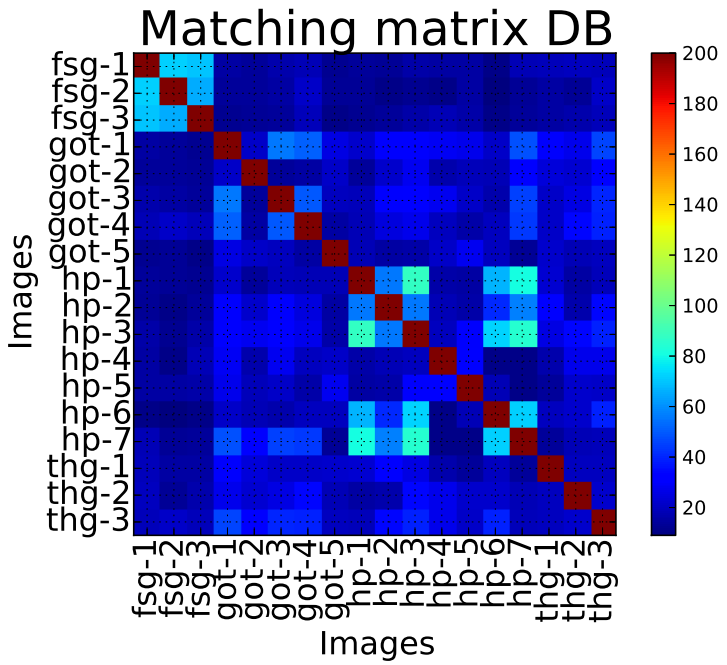
Figura 4.19: Ejemplo de emparejamientos entre portadas de la misma saga con el título de la saga en común.

La segunda prueba es una generalización de la anterior, realizando el emparejamiento con todas las portadas de la base de datos. Limitaremos el número máximo de emparejamientos a 200 y los representaremos en forma de matriz, obteniendo el resultado que se presenta en la Figura 4.20. Es de esperar que la matriz sea simétrica, por tanto podríamos crear una matriz triangular superior y posteriormente transponerla para completar la matriz triangular inferior. Para comprobar la repetibilidad en los resultados, en lugar de realizar el enfoque anterior, calcularemos el número de emparejamiento en las dos direcciones. Si definimos un par como $I_{ENTRADA} - I_{REFERENCIA}$, para las imágenes I_1 e I_2 tendremos dos direcciones, $I_1 - I_2$ e $I_2 - I_1$.

En primer lugar hemos de destacar que ambos algoritmos, SIFT y SURF, son suficientemente descriptivos para reconocer la imagen de entrada entre las disponibles en la base de datos. Esta conclusión la obtenemos observando la diagonal granate, lo que indica que el número de puntos de interés emparejados en el par $I_i - I_i$, para i=1..18, es muy alto. La principal diferencia que podemos observar entre ambos algoritmos es la aparición de patrones cuadrados cerca de la diagonal granate en los resultados de la Figura 4.20a. Analizando estos patrones se comprueba que corresponden a portadas de libros pertenecientes a la misma



(a) SURF



(b) SIFT

Figura 4.20: Número de emparejamientos cruzados para todas las imágenes de la BBDD1. Destacar la diagonal granate que representa el buen reconocimiento de las portadas y la aparición de patrones cuadrados que representan relaciones entre portadas.

saga, lo que parece razonable ya que las portadas tienen elementos en común: nombre del autor en la saga *Game of Thrones*, o título de la saga en las de *Harry Potter*. Obsérvese que el algoritmo SIFT, cuyos resultados se muestran en la Figura 4.20b, no detecta estos patrones comunes con la misma claridad que el algoritmo SURF.

4.3.3. Análisis BBDD-2

El objetivo del análisis con esta base de datos es evaluar las prestaciones de la aplicación de reconocimiento de portadas de libros utilizando una secuencia de vídeo en un entorno real. Al ser una aplicación de vídeo necesitamos que el tiempo de computación sea bajo, por lo que únicamente analizaremos esta característica. Nótese que el tiempo de computación mostrado en la Figura 4.21 corresponde a imágenes de la base de datos, y por tanto será diferente al obtenido al trabajar con imágenes captadas con una cámara de vídeo convencional (gama media-baja). Es razonable asumir que las imágenes de entrada serán similares a las de la base de datos y por tanto se podrán extrapolar los resultados.

Figura 4.21: Tiempo de detección, descripción, emparejamiento, tiempo total y número de puntos de interés para la BBDD-2 usando SIFT y SURF.

En la Figura 4.21 se observa que el tiempo de cómputo de SIFT es menor que el correspondiente a SURF. El único problema podría ser el tiempo dedicado a los emparejamientos, ya que este tiempo es mayor en SIFT que en SURF para la imagen de entrada *dio*. Para analizar este hecho se obtendrá la máxima

diferencia de tiempo dedicado a emparejamiento para SIFT y SURF. Para ello emparejamos la imagen de entrada con todas las imágenes de la base de datos, obteniendo que la mayor diferencia en tiempo de emparejamiento entre SIFT y SURF es 0.005 milisegundos. Por tanto la máxima diferencia de tiempo dedicado a emparejamiento será $6 \times 0.005 = 0.03$ milisegundos, que no afectará en gran medida al tiempo de cómputo total representado en la cuarta fila de la Figura 4.21. Remarcamos aquí que para el análisis del tiempo de emparejamiento se han usado dos *descriptores* iguales. No obstante, en una aplicación real estos *descriptores* serán distintos y probablemente el tiempo de emparejamiento aumente.

4.3.4. Análisis BBDD-3

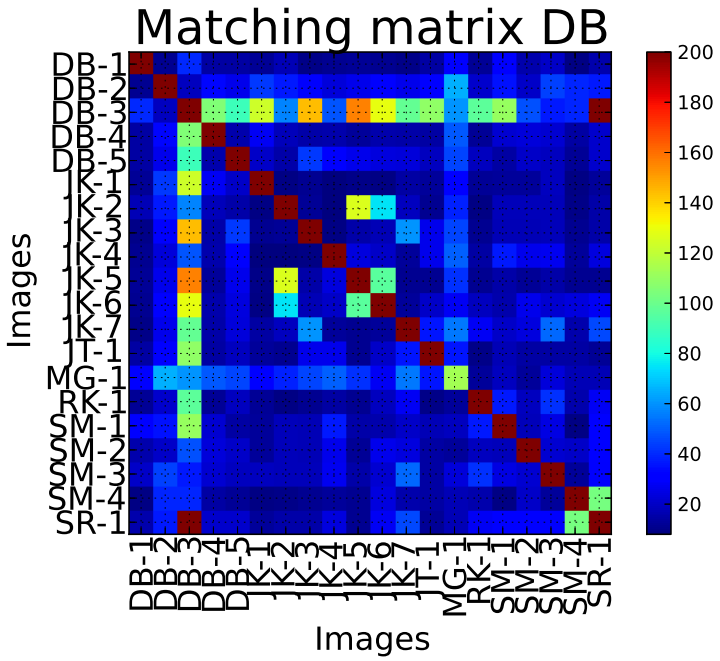
Esta base de datos se usará para ilustrar la aplicación de reconocimiento de objetos en imágenes estáticas. Decidiremos que algoritmo reconoce las portadas de forma más diferenciada, calculando la matriz de emparejamientos entre las diferentes imágenes que componen la base de datos, y analizando los resultados.

En la Figura 4.22 se muestra el número de emparejamientos, limitándolos a 200, para los algoritmos SIFT y SURF. En primer lugar, podemos observar que ambos algoritmos son suficientemente descriptivos para reconocer la imagen de entrada de entre las disponibles en la base de datos (diagonal granate de las Figuras 4.20a y 4.22b). La portada *MG-1* no muestra un número de emparejamientos alto porque su tamaño es muy reducido (143×93) y el número de puntos de interés detectados es menor que 200.

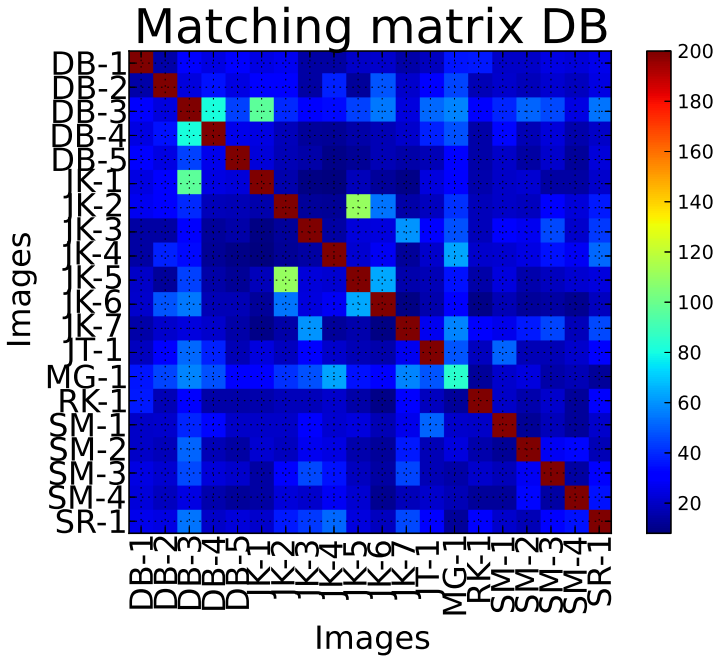
Para el algoritmo SURF, la portada *DB-3* muestra un alto número de emparejamientos con otras portadas, entre ellas *JK-5* y *SR-1* (véase Figura 4.22a). El número de puntos de interés detectados para la imagen *DB-3* es muy alto, debido al tamaño de la imagen (1186×702), y tenemos un gran número de "emparejamientos falsos". Como comentamos para la BBDD1, SIFT describe los puntos de interés de forma más específica que SURF, solventando este problema (véase Figura 4.22b).

Por tanto, se puede concluir que el tamaño de las imágenes de la base de datos puede afectar negativamente al proceso de reconocimiento. A continuación se proponen algunas soluciones para este problema:

- Modificar el tamaño de las imágenes para que todas las imágenes tengan el mismo tamaño. No usaremos este enfoque porque tendríamos que reducir todas las imágenes al tamaño de la más pequeña, con la pérdida de información que este cambio puede conllevar.
- Usar una medida distinta del número total de emparejamientos, que tenga en cuenta el número total de puntos de interés extraídos para cada imagen.
- Usar el descriptor SIFT. Es la opción escogida en este trabajo.



(a) SURF



(b) SIFT

Figura 4.22: Número de emparejamientos cruzados para todas las imágenes de la BBDD-3. Destacar los resultados para las portadas DB-3 y MG-1.

Capítulo 5

Aplicaciones

En los capítulos anteriores introdujimos al lector a la *visión artificial*, definimos los objetivos de este proyecto y las herramientas utilizadas. También describimos los conceptos generales y, de forma más detallada, explicamos y analizamos los dos algoritmos para el reconocimiento de objetos en imágenes tratados en este proyecto. En este capítulo describimos la solución propuesta y desarrollada para cada una de las aplicaciones de reconocimiento realizadas. En la Sección 5.1 mostramos algunos ejemplos sencillos de detección, clasificación y reconocimiento de objetos (libros y cuadros). La solución propuesta para el reconocimiento de objetos en imágenes estáticas se explica en la Sección 5.2. Comenzamos con un diagrama de las etapas seguidas, e ilustramos cada una de ellas con un ejemplo. De manera análoga explicamos el proceso seguido para el reconocimiento de objetos en vídeo, es decir, secuencias de imágenes, Sección 5.3, y terminamos explicando, en la Sección 5.4, la solución propuesta para trasladar la aplicación local a una aplicación web, lo que aumentaría la utilidad de las aplicaciones y facilitaría su uso.

5.1. Detección, clasificación y reconocimiento

La primera aplicación que podemos realizar es la *detección de objetos* en una imagen, para lo que se dispone de dos imágenes: la del objeto y la imagen de entrada sobre la que buscar el objeto, de mayor tamaño que la anterior. Este no es el objetivo fundamental del proyecto final de carrera, y por lo tanto no se han añadido módulos de preprocesado de la imagen o utilizado técnicas específicas para detección. *Sliding window* o *windowing* [49] es una técnica convencional para la detección de objetos en imágenes. Consiste en desplazar la imagen del objeto por la imagen de entrada que representa la escena. Para cada deplazamiento se

obtiene una puntuación que indica si la región de la escena sobre la que se efectúa la comparación corresponde con el objeto.

En nuestro proyecto podríamos adaptar la técnica de *sliding window* comparando, en lugar de la intensidad de los píxeles (imagen del objeto e imagen de una región de la escena), los *descriptores* extraídos para cada una de ellas. En los ejemplos mostrados a continuación no utilizaremos dicha técnica, extraeremos los *descriptores* para la imagen del objeto y la escena completa y los compararemos.

En las Figuras 5.1 y 5.2 se muestran ejemplos del uso de los detectores y descriptores para detectar objetos en imágenes. Las imágenes de mayor tamaño (izquierda y derecha) se corresponden con las imágenes de los objetos a detectar, y la imagen central es la imagen de entrada con una composición de distintas portadas sobre la que se desea detectar/localizar una portada concreta (objetos de interés). Los recuadros muestran el objeto detectado.

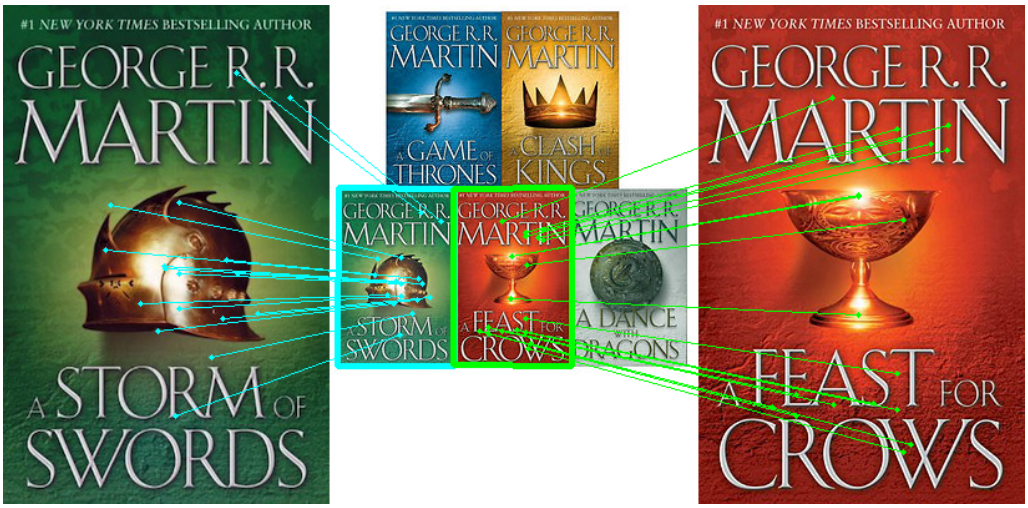


Figura 5.1: Ejemplo de detección de portadas de libros de la saga *Game of Thrones*. El descriptor usado es SURF.

Hemos de destacar que se ha tratado a la imagen central como un todo, emparejando todos los puntos de interés de la imagen central con los puntos de interés de las imágenes del objeto a detectar. Si realizamos un proceso de segmentación para separar cada portada de la imagen central y comparamos individualmente los puntos de interés de cada portada, estaríamos abordando un problema de reconocimiento. Es decir, en detección se busca la portada en una región de la imagen mientras que en reconococimiento se comparan dos portadas y se decide si corresponden al mismo objeto.

El siguiente problema que se puede afrontar es la *clasificación de objetos*: para una imagen de entrada con un objeto, recuperar de una base de datos los objetos con características visuales similares. En el ejemplo implementado consideraremos una parte de la base de datos de OpenLibrary con las 150 primeras portadas, y el

Figura 5.2: Ejemplo de detección de portadas de libros de la saga *Harry Potter*. El descriptor usado es SIFT.

número de emparejamientos entre dos imágenes como criterio de clasificación. En la Figura 5.3 se muestra el resultado de la recuperación de imágenes con elementos en común en dicha base de datos. Las imágenes están ordenadas de izquierda a derecha por mayor número de emparejamientos. En la fila superior está la imagen de entrada, con el título *Richard II*, y en la fila inferior las imágenes recuperadas de la base de datos con los emparejamientos realizados.

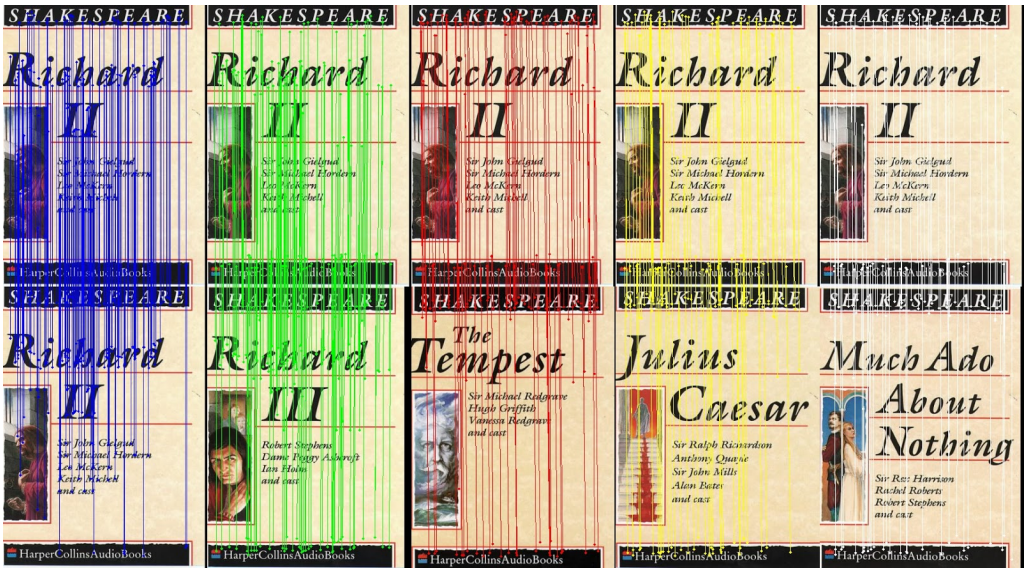


Figura 5.3: Ejemplo de clasificación y recuperación de imágenes con elementos comunes para la base de datos BBDD4 (150 portadas). El descriptor usado es SURF.

En primer lugar destacamos los elementos en común en todas las portadas, es decir, el autor y la editorial. Para las dos primeras imágenes recalcamos la similitud entre los títulos de los libros *Richard II* y *Richard III*. Podemos concluir

que la recuperación ha sido correcta, devolviéndonos, en primer lugar, la misma imagen que teníamos a la entrada (*Richard II*), en segundo lugar el siguiente volumen de la saga (*Richard III*) y para finalizar otras portadas del mismo autor y editorial con una distribución muy parecida. Nótese la ausencia de la portada *Richard I*, cuyo número de emparejamientos se intuye alto, debido a que no está disponible en la base de datos.

Por último, podemos centrarnos en *reconocer el objeto* contenido en la imagen de entrada. Éste es el propósito del proyecto, y por ello en las Secciones 5.2 y 5.3 se lleva a cabo un procesado previo para mejorar los resultados. El ejemplo que mostraremos a continuación es un ejemplo sencillo de reconocimiento de objetos sin procesado previo. Nótese que, aunque no hemos extraído el objeto de la escena para compararlo de forma independiente, consideramos este problema como de reconocimiento porque la mayor parte de puntos de interés detectados corresponderán con el objeto.

Puesto que en el Capítulo 2 se consideró como posible aplicación el reconocimiento de cuadros en un museo, en la Figura 5.4 se muestra un ejemplo sencillo de dicha aplicación. La imagen del cuadro se obtiene mediante la grabación de vídeo realizada con una cámara web. Para cada *frame* se extraen los puntos de interés y se comparan con los almacenados en una base de datos. En la Figura 5.4 se muestra la imagen almacenada en la base de datos (izquierda) y la imagen con el cuadro, imagen capturada en un escenario real (derecha).

Figura 5.4: Ejemplo de reconocimiento del cuadro *Magna Carta* (izquierda) y su localización en la escena (derecha). Se han usado detectores y descriptores SURF.

Nótese que entre la imagen tomada por la cámara web (derecha) y la imagen de la base de datos (izquierda) hay cambios de escala, iluminación, diferencia de calidad y transformaciones geométricas.

En las siguientes secciones se presentan los pasos a seguir para el reconocimiento de objetos en dos situaciones distintas. En el primer escenario dispondremos de una imagen estática (previamente adquirida) en la que podrán aparecer varios objetos. En el segundo escenario trabajaremos con un flujo continuo de imágenes adquiridas mediante una cámara web. Se mostrará el resultado obtenido tras el proceso de reconocimiento para cada *frame* del flujo de imágenes.

5.2. Reconocimiento de objetos en imágenes estáticas

El objetivo principal de esta aplicación es reconocer portadas de libros o cualquier otro tipo de objeto presente en una imagen. El esquema de reconocimiento se muestra en la Figura 5.5, donde el bloque de *Procesado* del diagrama general (Figura 3.1) presentado en el Capítulo 3, se ha concretado en una etapa de *Segmentación*. Dicha etapa obtendrá los objetos de interés, que posteriormente se compararán con los almacenados en una base de datos.

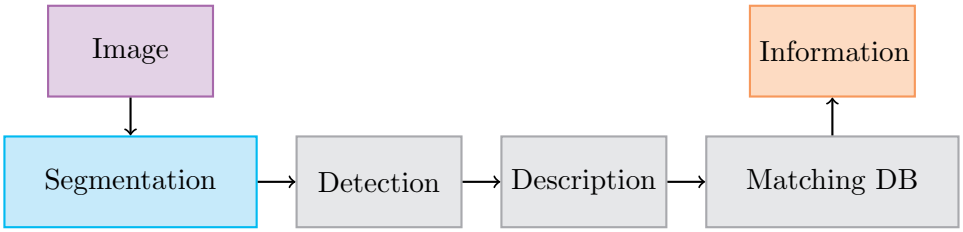


Figura 5.5: Diagrama de reconocimiento de objetos en imágenes estáticas. El procesado de imagen se concreta en una etapa de segmentación para localizar los objetos de interés.

A continuación se explica cada bloque del diagrama de la Figura 5.5 y se ilustra con un ejemplo.

A) Imagen

La imagen de entrada al sistema de reconocimiento puede contener uno o varios objetos, como muestra la Figura 5.6. Recordemos que el color está definido por dos magnitudes, la saturación, que indica la viveza del color, y el tinte, que indica la tonalidad del color. El *background* de la Figura 5.6 muestra distintos tintes, blanco en las letras y un rango de marrones, con diferentes magnitudes de saturación en la estantería. Además, aparece una textura no homogénea, imitando el vetado de la madera y sus variaciones debido a la iluminación. Por estos motivos consideraremos que el *background* de la imagen es complejo. Los objetos de la Figura 5.6 a reconocer son libros.

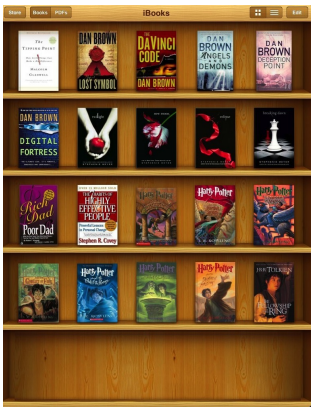
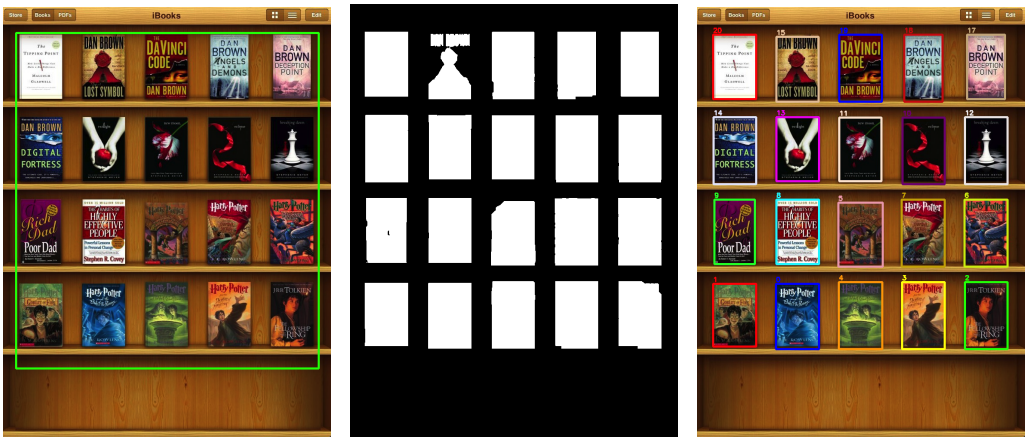


Figura 5.6: Imagen de entrada al sistema de reconocimiento.

B) Segmentación

Para el proceso de segmentación se ha utilizado el algoritmo de *Grabcut*. Recordemos que este algoritmo es el más complejo de los presentados en el Capítulo 3 y necesita que el usuario indique inicialmente una región de interés, definida por el rectángulo verde de la Figura 5.7a. Los píxeles que yacen en el rectángulo se considerarán *foreground* y el resto *background*. Una vez indicada la región de interés, el algoritmo de *Grabcut* estimará el *background* y proporcionará como resultado una imagen binaria, Figura 5.7b: a los píxeles pertenecientes al *background* (color negro) se les asignará el valor 0, y a los píxeles relativos al *foreground* el valor 255.



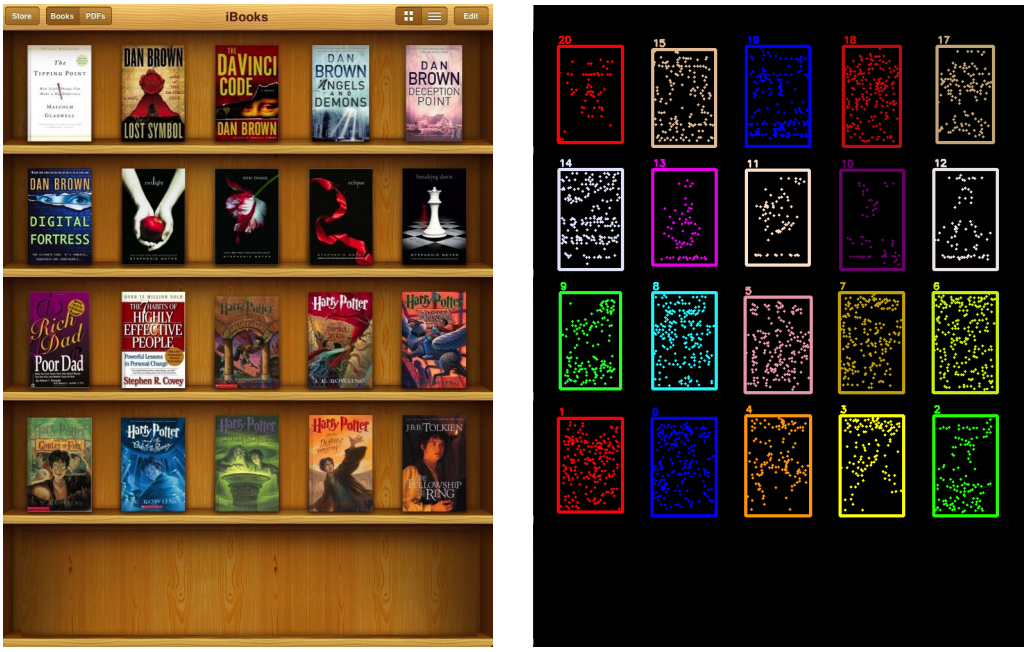
(a) Segmentación inicial (b) Resultado *Grabcut* (c) *Bounding boxes*

Figura 5.7: Ejemplo de segmentación y separación por componentes conexas para el reconocimiento de varios objetos en una misma imagen.

Para segmentar cada uno de los libros de la imagen se realiza un análisis por componentes conexas y se determina la *bounding box* para cada objeto. En la Figura 5.7c se muestra la *bounding box* para cada libro junto con un número que lo identifica. Puesto que el segundo libro de la primera fila está dividido en dos componentes conexas se generarían dos *bounding boxes* distintas. Para solventar esta dificultad, hemos definido una longitud mínima de anchura y altura para la *bounding box* de una componente conexa. La componente conexa será considerada como un objeto independiente si supera dichas dimensiones mínimas.

C) Detección y descripción

A partir de la *bounding box* obtenida para cada objeto se extrae la región correspondiente de la imagen de entrada, se detectan los puntos de interés (véase la Figura 5.8b) y se calculan sus *vectores descriptores*, obteniendo un *descriptor* para cada objeto. El algoritmo de detección y descripción empleado es SIFT. En el análisis de las bases de datos del Capítulo 4 se mostró que, para portadas con elementos comúnes, SIFT las caracteriza/describe de forma más diferenciada que SURF y por tanto se adecúa a nuestro objetivo.



(a) Imagen de entrada (b) Puntos de interés

Figura 5.8: Puntos de interés detectados para cada objeto hallado en la imagen de entrada. El detector y descriptor usado es SIFT.

D) Emparejamiento

Tenemos los *descriptores* de los objetos que aparecen en la imagen y queremos identificarlos con los objetos de una base de datos. Para ello compararemos cada *descriptor* de un objeto con los *descriptores* almacenados en la base de datos. El proceso de selección de los emparejamientos correctos entre dos *descriptores* se puede realizar de muchas maneras; los pasos seguidos en este ejemplo se explican a continuación:

- Cada emparejamiento realizado entre dos puntos de interés, tiene asociado una distancia euclídea. Recordemos que cada punto de interés viene descrito por un *vector descriptor*. En primer lugar ordenamos los emparejamientos por distancia.
- Seleccionamos los N mejores emparejamientos, es decir, los N emparejamientos con menor distancia euclídea. En nuestro caso hemos fijado *N=100*.
- Para los emparejamientos seleccionados, buscamos la transformación correspondiente entre el conjunto de puntos de interés del objeto, y el conjunto de puntos de interés de la imagen de la base de datos. Esto se hace mediante el cálculo de su homografía, eliminando de este modo aquellos emparejamientos que no cumplen dicha transformación y por tanto son considerados *outliers*.

Tras realizar estos pasos, obtendremos un vector con el número de emparejamientos correctos entre el objeto de entrada y cada uno de los objetos de la base de datos. Finalmente seleccionaremos el objeto de la base de datos con mayor número de emparejamientos correctos. Este proceso se repite para todos los objetos de la imagen.

E) Información

Finalmente mostraremos la información relativa a cada objeto. En este caso se presenta el título del libro sobre la imagen de entrada, como se ilustra en la Figura 5.9. Nótese que las dos letras en mayúscula corresponden con las iniciales del autor, y despues le sigue una palabra perteneciente al título. Por ejemplo, *DB-angels* corresponde con el libro *"Dan Brown - Ángeles y demonios"* o *JK-hp-7* corresponde con el libro *"J.K. Rowling - Harry Potter and the deathly hallows"* que es el séptimo libro de la saga.

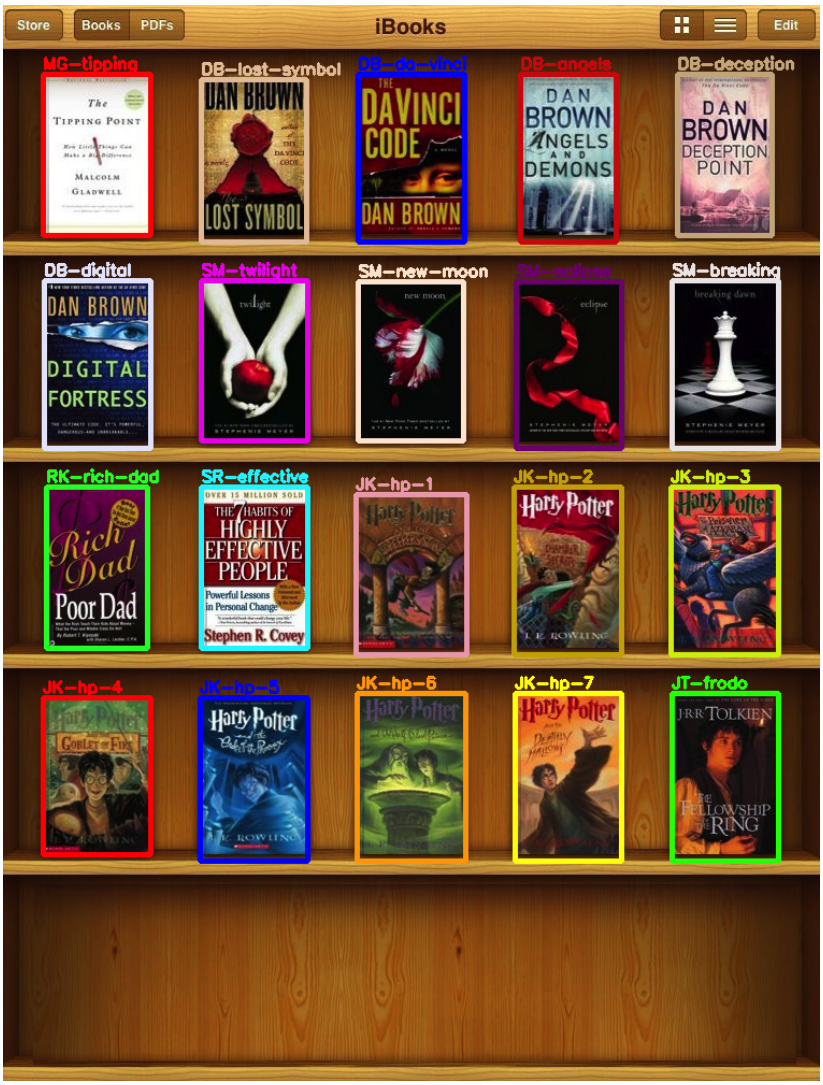


Figura 5.9: Información mostrada para cada uno de los objetos reconocidos tras haber sido identificados con su mejor emparejamiento de la base de datos.

5.3. Reconocimiento de objetos en vídeo

El reconocimiento de objetos en secuencias de imágenes se efectúa de forma similiar al reconocimiento de objetos en imágenes estáticas (el vídeo es un flujo continuo de imágenes). La principal diferencia reside en la necesidad de ejecutar los algoritmos en tiempo real[1] para no romper la sensación de "fluidez visual" del vídeo. Por este motivo es necesario que el tiempo de computación entre imágenes (*frames*) consecutivos sea lo menor posible. Para conseguir esta mejora en el

[1]Un sistema de tiempo real, es aquel en el que se establecen unas restricciones temporales para la obtención de los resultados.

tiempo de computación se pueden seguir varios enfoques, algunos de los cuales son:

A) Selección de algoritmos de reconocimiento computacionalmente eficientes

Aunque hay algoritmos más rápidos que SURF y SIFT, la relación entre calidad de los resultados y tiempo de computación hace decantarnos por ellos, tal y como se indica en los análisis del Capítulo 4.

B) Reducción del número de puntos de interés detectados

En el Capítulo 4 se comprobó que el tiempo de cómputo global dependía principalmente del tiempo dedicado a la descripción. Una reducción del número de puntos de interés detectados disminuirá el número de puntos a describir y por tanto el tiempo dedicado a esta tarea. Como se explica más adelante, centraremos nuestros esfuerzos en este enfoque para la reducción del tiempo de computación.

C) Búsqueda eficiente en la base de datos

Uno de los aspectos para que la aplicación funcione en tiempo real es el emparejamiento entre la imagen de entrada y las imágenes almacenadas en la base de datos. Si el número de elementos almacenados en la base de datos es muy alto, el emparejamiento será un punto crítico. El enfoque ideal para solucionar este problema es usar estadísticas de la base de datos para saber dónde buscar en primer lugar. Supongamos que la imagen del usuario es un zapato de tacón de color amarillo, y en la base de datos tenemos muchos zapatos de tacón, pero muy pocos amarillos, la idea sería comenzar a buscar por el color amarillo. Obviamente, este es otro problema distinto al abordado en este proyecto, por lo que las soluciones aplicadas en este trabajo son muy básicas. La primera opción es tener una base de datos reducida para aplicaciones en tiempo real. La segunda es no recorrer siempre toda la base de datos y utilizar un umbral de calidad (número de emparejamientos correctos) a partir del cuál considerar que la imagen de entrada y la imagen de la base de datos coinciden, deteniendo así la búsqueda en la base de datos.

D) Uso de programación paralela y distribuida

La solución distribuida es muy compleja y tiene limitaciones debido a la necesidad de comunicación entre los diferentes ordenadores/servidores. Las aplicaciones presentadas en este proyecto consideran tiempos muy bajos, por lo que una solución de este tipo podría afectar negatívamente. Si una base de datos de gran tamaño se distribuye en varios servidores, los tiempos de comunicación entre dispositivo y servidores serán despreciables en comparación con la reducción de tiempo conseguida por la comparación paralela de la imagen de entrada con las imágenes de la base de datos. Si por el contrario se

trabaja con bases de datos de tamaño reducido, los tiempos de comunicación podrían resultar un problema.

Por otro lado, OpenCV incluye un módulo GPU, mantenido por NVIDIA, con código acelerado para gran parte de las funciones de la librería, proporcionando compatibilidad con las GPU de Intel. El módulo, escrito en CUDA, está siendo adaptado a las nuevas tecnologías de programación y arquitecturas GPU. El cálculo de los puntos de interés del algoritmo SURF usando la GPU es 12 veces más rapido que usando la CPU, como indican en OpenCV[2]. El principal problema es la compatibilidad del código entre distintas GPUs.

El objetivo principal de la aplicación presentada en esta sección es reconocer portadas de libros (extensible a cualquier otro tipo de objeto) en una secuencia de imágenes. El esquema de reconocimiento se muestra en la Figura 5.10, donde el bloque de *Procesado* del diagrama general (Figura 3.1) presentado en el Capítulo 3, se ha concretado en una etapa de *Sustracción de fondo*. Como resultado de dicha etapa se obtendrá el objeto de interés que posteriormente se comparará con los almacenados en una base de datos.

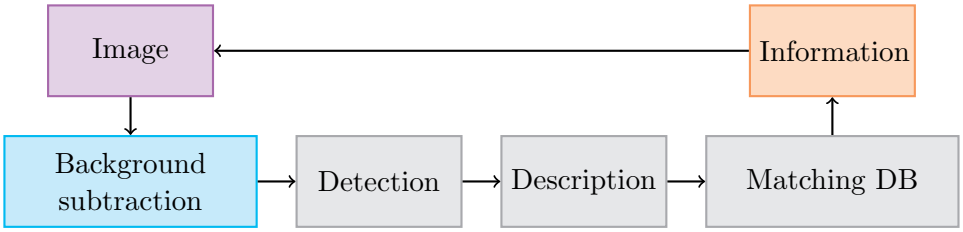


Figura 5.10: Diagrama de reconocimiento de objetos en vídeo. Previo a la detección de puntos de interés se realiza una sustracción de fondo para detectar el objeto a reconocer.

A continuación se explica cada bloque del diagrama de la Figura 5.10, ilustrando el resultado de cada bloque con un ejemplo.

A) Imagen

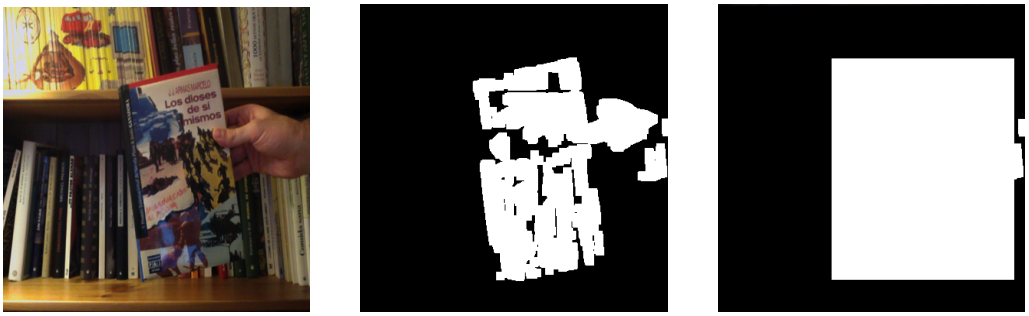
La imagen de entrada al sistema de reconocimiento ha de contener como máximo un objeto. En la Figura 5.11a se muestra una posible imagen de entrada tomada con una cámara web. Los objetos a reconocer en nuestro caso son libros, y hemos elegido un *background* complejo.

B) Sustracción de fondo

Para el proceso de sustracción de fondo, éste se ha modelado a través de un esquema *Mixture of Gaussians (MoG)*. Recuérdese que disponemos de un

[2]Link Speedup GPU: http://opencv.org/platforms/cuda.html

parámetro que indica la tasa de aprendizaje, es decir, la rapidez con la que la estimación se adecúa a los cambios en el fondo. Así, si nos encontramos ante un fondo constante, la tasa de aprendizaje necesaria será baja; si por el contrario el fondo varía, aumentaremos el valor de la tasa de aprendizaje para modelar con mayor rapidez los cambios. Remarcar que si la tasa de aprendizaje es alta y el objeto de interés, en nuestro caso el libro, se sitúa estáticamente delante de la camára de vídeo, en los siguientes *frames* el libro pasará a considerarse parte del fondo. Por lo tanto, interesa utilizar tasas de aprendizaje bajas para considerar el libro como primer plano, aún cuando éste esté inmóvil delante de la camára.



(a) Imagen de entrada (b) Resultado *MoG* (c) *Bounding boxes*

Figura 5.11: Ejemplo de sustracción de fondo para el reconocimiento de objetos en vídeo.

Tras aplicar el algoritmo *MoG* obtendremos una imagen binaria, Figura 5.11b. A los píxeles pertenecientes al fondo (color negro) se les asignará el valor 0, y el valor 255 a los píxeles relativos al objeto de interés. Vemos que la aproximación obtenida para el objeto de interés es buena, pero aún así hay zonas del objeto que son consideradas como fondo. Para extraer puntos de interés de la totalidad del objeto se realiza un análisis por componentes conexas y se determina su *bounding box*. En la Figura 5.11c se muestra el resultado tras considerar toda la zona relativa al *bounding box* como objeto de interés.

C) Detección y descripción

Tras extraer la región asociada al objeto de interés a través de una estimación de su *bounding box*, comienza la etapa de detección de puntos de interés y el cálculo de sus vectores descriptores. El resultado se muestra en la Figura 5.12. El algoritmo de detección y descripción empleado es SURF.

D) Emparejamiento

Se dispone del *descriptor* del objeto de interés y queremos identificar el objeto de la base de datos correspondiente. Para ello compararemos el *descriptor*

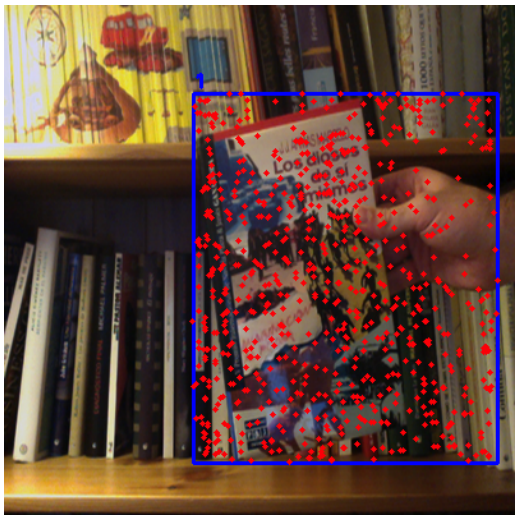


Figura 5.12: Puntos de interés detectados con SURF para el objeto de interés.

del objeto con los descriptores almacenados en la base de datos. El proceso de selección de los emparejamientos correctos entre dos puntos de interés se puede realizar de muchos modos; a continuación se explican los pasos seguidos en este ejemplo.

- Cada emparejamiento entre dos puntos de interés (descritos por sus *vector descriptors*) tiene asociado una distancia euclídea. Los emparejamientos se ordenan por distancia creciente.
- Se seleccionan los N mejores emparejamientos, es decir, los N emparejamientos con menor distancia euclídea. En nuestro caso se ha fijado N=*100*.
- Para los emparejamientos seleccionados, se busca la transformación correspondiente entre el conjunto de puntos de interés del objeto y el conjunto de puntos de interés de la imagen de la base de datos. Esto se hace mediante el cálculo de su homografía, eliminando de este modo aquellos emparejamientos que no cumplen dicha transformación y por tanto son considerados *outliers*.

Tras estos pasos obtendremos el número de emparejamientos correctos entre el objeto de entrada y los de la base de datos. Si el número de emparejamientos supera un determinado umbral, consideraremos que el objeto de entrada y el objeto de la base de datos coinciden. Nótese que si el número de emparejamientos umbral se supera para la primera imagen a comparar, no será necesario comparar con el resto de elementos de la base de datos y el tiempo de cómputo se reducirá.

E) Información

Tras el emparejamiento de la etapa anterior se mostrará la información relativa al objeto obtenido. En esta aplicación, esa información corresponderá a la portada almacenada en la base de datos, junto con los emparejamientos y el título del libro, como se muestra en la Figura 5.13.

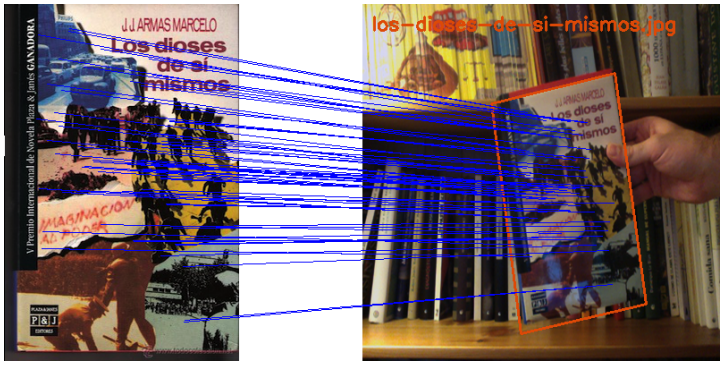


Figura 5.13: Información mostrada para el objeto tras haber sido reconocido.

5.4. Aplicación web

Las aplicaciones presentadas en este capítulo se han implementado y ejecutado de forma local en un ordenador personal. Para ser usadas en dispositivos móviles como teléfonos o tabletas se ha elegido como opción el paradigma de *Cloud Computing* mediante el uso del *framework* de desarollo web Django. Las razones que han motivado esta elección son las siguientes.

1. El desarrollo de la aplicación en un entorno web permitirá el uso de la misma por todos los sistemas operativos diseñados para móviles. En caso contrario tendríamos que desarrollar la aplicación web de forma particularizada para cada uno de ellos. PhoneGap o HTML5 pueden utilizarse para acceder al hardware de los dispositivos, en nuestro caso la cámara.

2. En las aplicaciones presentadas siempre se especificaba previamente el contenido de la base de datos. Téngase en cuenta que cada usuario debe poder crear sus propias bases de datos con los objetos a reconocer. Podemos elegir guardar dichas imágenes en el propio dispositivo móvil o, por el contrario, almacenarlas en un servidor común. Se ha considerado mejor opción almacenarlas en un servidor común, por motivos de capacidad y para que los usuarios puedan reusar/completar las bases de datos creadas por otros.

3. Como se mencionó previamente, este proyecto final de carrera ha sido desarrollado en Python. Es por tanto una opción muy buena el uso de Django,

basado en el mismo lenguaje de programación. La combinación de Django con las librerías de Python como Matplotlib proporciona una gran versatilidad al desarrollador.

Capítulo 6

Conclusiones

En los capítulos anteriores se ha descrito el esquema de reconocimiento propuesto, se han realizado experimentos para seleccionar el algoritmo que más se ajuste a las necesasidades de las aplicaciones y se han mostrado ejemplos que validan el sistema desarrollado. En este último capítulo se exponen las principales conclusiones obtenidas de este proyecto fin de carrera y las posibles líneas futuras para mejorar el sistema y en las que se puede seguir investigando, con lo que se cierra la presente memoria.

6.1. Conclusiones

La principal conclusión tras la realización de este proyecto final de carrera es, sin duda alguna, que la visión por computador es un campo en constante investigación, y que requiere gran cantidad de conocimientos. Estos conocimientos pueden ir desde conceptos básicos en tratamiento de imágenes, como cambios de iluminación, escalado y transformaciones a otros espacios (transformada de Fourier, transformada de Hough) hasta conceptos avanzados de segmentación, teoría de grafos y extracción de características, por citar algunos ejemplos. A continuación se presentan algunas de las conclusiones obtenidas tras la realización de este proyecto.

- Los creadores del descriptor SURF destacan que su algoritmo es varias veces más rápido que el anteriormente desarrollado SIFT. De nuestros análisis, podemos concluir que esta característica no se mantiene en las implementaciones de dichos descriptores en OpenCV. Nótese la importancia de estudiar y verificar las características de las implementaciones usadas, en este caso implementaciones libres.

- Tras los análisis realizados podemos concluir que para las transformaciones de escalado, cambio de iluminación, difuminado y rotación, los descriptores SIFT y SURF ofrecen buenos resultados.
- La elección del descriptor "ideal" no es una tarea sencilla. En este trabajo, las elecciones se han realizado en base a las aplicaciones finales y a las propias imágenes de la base de datos. Los resultados muestran que el descriptor SIFT detecta y describe los puntos de interés de manera más singular que el descriptor SURF, siendo por tanto el primero usado para abordar tareas de reconocimiento. Por el contrario, el descriptor SURF será usado para tareas de clasificación.
- La detección de puntos de interés en zonas que no pertenecen al primer plano (objeto de interés) aumenta el tiempo de cómputo y el número de falsos positivos, empeorando el proceso de reconocimiento. La solución propuesta modela el fondo para determinar los píxeles correspondientes al primer plano y evitar este problema.
- En el reconocimiento de N objetos en una misma imagen surgen varios problemas. Por un lado, el tiempo dedicado al emparejamiento aumenta porque se comparan los puntos de interés de los N objetos con cada imagen de la base de datos. Además, al comparar puntos de interés de diferentes objetos también se incrementa el número de falsos positivos. La solución aquí propuesta segmenta la imagen de entrada, obteniendo de manera independiente los puntos de interés de cada objeto para su posterior emparejamiento.
- Hay muchos criterios para determinar si el *descriptor* de un objeto se corresponde con el *descriptor* de otro objeto en una base de datos. El mejor criterio a considerar en cada caso dependerá de las propiedades de las imágenes de la base de datos, tamaño y contenido de las mismas.

A) Conocimientos aplicados

Los conocimientos adquiridos en la carrera y aplicados durante la realización del proyecto final de carrera son más generales que concretos. En la carrera se han estudiado numerosos lenguajes de programación y se han aprendido distintos paradigmas de programación. El lenguaje de programación usado en este proyecto ha sido Python, ya estudiado en la carrera y que por tanto no ha sido necesario aprender desde cero.

Las asignaturas más relacionadas con este proyecto final de carrera son, sin duda, las dos asignaturas de Tratamiento Digital de Señales Multimedia. De ellas hemos usado los conocimientos básicos en tratamiento de imágenes, iluminación, escalado, difuminado y detección. También hemos aplicado conocimientos de *machine learning* o aprendizaje máquina, un ejemplo de ello son los algoritmos K-NN y K-medias.

B) Conocimientos adquiridos

Destacar que, excepto por los conocimientos básicos adquiridos en las asignaturas de Tratamiento Digital de Señales Multimedia, las nociones de reconocimiento de objetos en imágenes eran mínimas. La mayor parte del contenido, desde conceptos relacionados con puntos de interés y su caracterización, o la sustracción de fondo (explicados en el capítulo de *Contexto*), han sido estudiados durante el desarrollo del proyecto final de carrera.

Hemos mejorado notablemente los conocimientos de programación en Python y aprendido a usar algunas de sus librerías externas, como Matplotlib, Numpy o SciPy, que han demostrado ser sin ninguna duda de muchísima utilidad. Ha de hacerse mención especial del uso de la librería OpenCV.

También hemos desarrollado nuestra habilidad para buscar soluciones a los problemas a los que nos hemos enfrentado, por ejemplo el uso de segmentación para separar objetos pertenecientes al primer plano de la imagen, o la sustracción del fondo en la aplicación de vídeo. También ha mejorado nuestra comprensión a la hora de leer documentos escritos por otros investigadores, analizarlos y tomar decisiones sobre su utilidad para nuestras necesidades.

Al comenzar el proyecto final de carrera ya disponíamos de conocimientos básicos de LaTeX[1], debido a su uso para la redacción de algunas memorias durante la carrera. Estos conocimientos se han ampliado notablemente al redactar este documento y serán de mucha utilidad para la elaboración de futuros escritos.

C) Uso de software libre

En este proyecto se ha intentado fomentar, en la medida de lo posible, el uso de software libre. Prueba de ello es la elección de OpenCV como librería base para el tratamiendo de imágenes, la elección de Django como *framework* de desarrollo web o el uso de imágenes de OpenLibrary y su API. Durante la realización de este proyecto se ha comprendido la importancia de este tipo de software para fomentar el desarrollo libre y facilitar la comunicación y el uso de conocimientos.

6.2. Líneas futuras

Las mejoras que se pueden realizar a los algoritmos son inumerables y de hecho, al ser un área de reciente investigación, muchas de ellas probablemente serán desarrolladas en un futuro. Entre las mejoras que consideraría relevantes caben destacar:

[1]Sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos. Creado por Leslie Lamport en 1984, separa las macros de estilo del contenido.

A) En primer lugar aumentar la robustez frente a transformaciones geométricas afines, ya que éstas serán, sin lugar a dudas, una de las transformaciones que más frecuentemente apareceran en la imagen que el usuario tome con un dispositivo móvil. Métodos como el desarrollado por Jean Michel Morel y Guoshen Yu, denominado ASIFT [41], podrían ser efectivos aunque también aumentarían el número de falsos positivos debido al incremento (puede que excesivo) de puntos de interés.

B) Dado que cada usuario creará su propia lista con las imágenes que desea reconocer, sería interesante particularizar los parámetros de los algoritmos para cada lista. Esto se puede conseguir aplicando un esquema de *cross validation* que considere distintos parámetros, de modo que se elijan aquéllos que ofrezcan mejores prestaciones. Existen métodos más complejos desarrollados específicamente para bases de datos con varias vistas del mismo objeto, por ejemplo creando ejemplos de forma virtual [50].

C) Agrupar los puntos de interés y sus vectores descriptores de forma lógica, para lo que podría usarse el enfoque desarrollado por Nister y Stewenius [45] y evaluado por Tomasik, Thiba y Turnbull [46]. La idea es agrupar las características extraídas previamente en un vocabulario de palabras visuales denominado *bag of words*, de modo que cada grupo de características represente una propiedad del objeto (por ejemplo, la cremallera o los cordones de unas zapatillas).

D) Extender el trabajo de reconocimiento de objetos a objetos con más de una vista canónica. Para ello se podría contemplar el método presentado por Rothganger et al [51], que representa la forma 3D del objeto considerando más de una vista canónica. Posteriormente la superficie se divide en pequeñas e invariantes zonas o parches, de las cuales se extraen los detectores y descriptores de modo similar al considerado en este trabajo, descriptores que también almacenarán las relaciones tridimensionales entre diversos parches. Para una explicación más detalla, véase *Toward True 3D Object Recognition* [52].

E) Estimar de forma probabilística el modelo 3D del objeto [53] podría ayudar a determinar qué vistas del objeto son más relevantes, así como el número de puntos de interés a extraer de cada vista [54].

F) Una búsqueda eficiente del objeto a reconocer en la base de datos. Un enfoque para reducir el tiempo de búsqueda consiste en usar las estadísticas almacenadas en la base de datos [55].

Apéndice A

Librerías y procedimientos

Para la realización de este proyecto hemos trabajado principalmente con *scripts*. Se han creado *scripts* para segmentación, sustracción de fondo, detección, emparejamiento y para cada uno de los tests realizados en el Capítulo 4, entre otros. Estos *scripts* se caracterizan por usar una serie de clases con métodos públicos a las que denominamos *handlers*. Estas clases, que hacen la función de librerías, serán explicadas a continuación.

A) handler-matches.py

Realiza el emparejamiento entre los puntos de interés de dos imágenes. Además permite filtrar los puntos de interés de diversas maneras, ordenándolos y seleccionando los N mejores, escogiendo aquéllos cuya distancia es menor que un cierto valor o recopilando sólo aquéllos que cumplen una determinada homografía.

- *match(descriptors, type)*
- *matches_idxs_to_keypoints(matches, kp-pairs)*
- *filter_matches_by_ratio(matches, ratio)*
- *filter_matches_by_distance(matches, nMatches)*
- *filter_matches_by_homography(matches, kp-pairs, ransac)*

B) handler-draws.py

Se encarga de representar los resultados. Tiene funciones para representar varias imágenes juntas, sus puntos de interés, líneas de emparejamientos y homografías.

- *figure(images, rows, cols)*

- *draw_Background(image-out, image-in, offsets, rcp, fgray)*
- *draw_Keypoints(image-out, keypoints, offsets, rcp, frad, color=None)*
- *draw_Matches(image-out, keypoint-pairs, offsets, rcp, color=None)*
- *draw_Homography(image-out, image-size, offsets, rcp, H, color=None)*

C) handler-extraction.py

Se encarga de cargar imágenes y calcular sus puntos de interés y vectores de descripción. Permite aplicar máscaras a la imagen para seleccionar las zonas de las que extraer los puntos de interés, y permite binarizar una imagen para separar primer plano y fondo.

- *__init__(detector-type, descriptor-type)*
- *compute_path(image-path, type-mask)*
- *compute_img(image, type-mask)*
- *get_mask(img, type-mask)*

D) handler-db.py

Se encarga de recorrer los elementos de un directorio para después calcular sus puntos de interés y vectores descriptores, almacenarlos y acceder a ellos de forma eficiente.

- *__init__(extractor, type-mask)*
- *load_database(db-path)*
- *parse_folder(folder-path)*
- *parse_file(file-path)*

Apéndice B

Interfaz

En el presente apéndice se explica con más detalle el método de uso y funcionamiento del interfaz de la aplicación de reconocimiento de cuadros que se muestra en la Figura B.1, realizada con el creador de prototipos *JustInMind*®.

Figura B.1: Ejemplo de la interfaz para la aplicación de cuadros.

En el recuadro definido por el marco se muestra la imagen de la base de datos del cuadro reconocido. La funcionalidad de los distintos botones de la aplicación es la siguiente:

- *YOUR information:* Al hacer clic en este botón se muestra la información introducida por el usuario para el cuadro reconocido.
- *WIKIPEDIA information:* Al hacer clic en este botón se muestra la información de Wikipedia para el cuadro reconocido.
- *Social comments:* Al hacer clic en este botón se muestran los comentarios de otros usuarios de la aplicación relativos al cuadro reconocido.
- *Newspapers comments:* Al hacer clic en este botón se muestran los comentarios realizados por periódicos relativos al cuadro reconocido.
- *Choose list:* Al hacer clic en este botón se muestran las listas de los diferentes museos. Tras seleccionar una lista, la imagen tomada será emparejada con las imágenes de la base de datos relativa a dicha lista.
- *Camera:* Permite realizar una foto del cuadro.

Bibliografía

[1] M. Šonka, V. Hlaváč, and R. Boyle. *Ise-Imgage Processing, Analysis and Machine Vision*. International student edition. Thomson Learning EMEA, Limited, 2008.

[2] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Always learning. Pearson Education, Limited, 2011.

[3] C. Steger, M. Ulrich, and C. Wiedemann. *Machine Vision Algorithms and Applications*. Wiley-VCH Textbook. Wiley, 2008.

[4] Vildan Tanriverdi and Robert J. K. Jacob. Interacting with eye movements in virtual environments. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '00, pages 265–272, New York, NY, USA, 2000. ACM.

[5] Robert J. K. Jacob. The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Trans. Inf. Syst.*, 9(2):152–169, April 1991.

[6] Fred Turek. Machine vision fundamentals. how to make robots see. *NASA Tech briefs magazine*, 35(6):60–62, 2011.

[7] Ra Lauric and Sarah Frisken. Soft segmentation of ct brain data [online], 2007. Last visited on 05/06/2013.

[8] YingLi Tian, R.S. Feris, Haowei Liu, A. Hampapur, and Ming-Ting Sun. Robust detection of abandoned and removed objects in complex surveillance videos. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(5):565–576, 2011.

[9] K. Lai, Liefeng Bo, Xiaofeng Ren, and D. Fox. Detection-based object labeling in 3d scenes. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1330–1337, 2012.

[10] J. Solem. *Programming Computer Vision with Python: Tools and Algorithms for Analyzing Images*. Oreilly and Associate Series. O'Reilly Media, Incorporated, 2012.

[11] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[12] S. Sonnenburg, G. Rätsch, S. Henschel, C.Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc. The SHOGUN machine learning toolbox. *Journal of Machine Learning Research*, 11:1799–1802, June 2010.

[13] J.C. Russ. *The Image Processing Handbook, Sixth Edition*. Taylor & Francis, 2011.

[14] W.K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. Wiley, 2007.

[15] A.C. Bovik. *The Essential Guide to Image Processing*. Elsevier Science, 2009.

[16] Stefan Carlsson. Geometric computing in image analysis and visualization [online], 2007. Last visited on 12/06/2013.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2012 (voc2012) results [online], 2012. Last visited on 15/06/2013.

[18] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[19] Y.Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112 vol.1, 2001.

[20] Ondrej Danek. Graph cut based image segmentation in fluorescence microscopy [online], 2008. Last visited on 11/06/2013.

[21] Christoph Göring, Björn Fröhlich, and Joachim Denzler. Semantic segmentation using grabcut. In *VISAPP (1)*, pages 597–602, 2012.

[22] Justin F. Talbot and Xiaoqian Xu. Implementing grabcut [online], 2006. Last visited on 19/06/2013.

[23] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland. Pfinder: real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):780–785, 1997.

[24] Li Guan. Background subtraction with grabcut [online], 2008. Last visited on 01/06/2013.

[25] Yannick Benezeth, Pierre-Marc Jodoin, Bruno Emile, Hélène Laurent, and Christophe Rosenberger. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, 19, July 2010.

[26] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages –252 Vol. 2, 1999.

[27] P. W. Power and J. A. Schoonees. Understanding background mixture models for foreground segmentation. In *Proceedings Image and Vision Computing New Zealand*, page 267, 2002.

[28] Jae Kyu Suhr, Ho Gi Jung, Gen Li, and Jaihie Kim. Mixture of gaussians-based background subtraction for bayer-pattern image sequences. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(3):365–370, 2011.

[29] P. L M Bouttefroy, A. Bouzerdoum, S.L. Phung, and A. Beghdadi. On the analysis of background subtraction techniques using gaussian mixture models. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4042–4045, 2010.

[30] Thierry Bouwmans, Fida El Baf, and Bertrand Vachon. Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey. *Recent Patents on Computer Science*, 1(3):219–237, November 2008.

[31] Edward Rosten, R. Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, 2010.

[32] Motilal Agrawal, Kurt Konolige, and MortenRufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, volume 5305 of *Lecture Notes in Computer Science*, pages 102–115. Springer Berlin Heidelberg, 2008.

[33] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 778–792. Springer Berlin Heidelberg, 2010.

[34] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555, 2011.

[35] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571, 2011.

[36] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, July 2008.

[37] Adam Schmidt, Marek Kraft, Michał Fularz, and Zuzanna Domagała. Comparative assessment of point feature detectors and descriptors in the context of robot navigation. *Journal of Automation, Mobile Robotics and Intelligent Systems*, 7(1):11–20, 2013.

[38] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[39] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[40] Yan Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506–II–513 Vol.2, 2004.

[41] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci.*, 2(2):438–469, April 2009.

[42] Tony Lindeberg. Scale-space theory: a basic tool for analyzing structures at different scales. *Journal of Applied Statistics*, 21(1-2):225–270, 1994.

[43] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

[44] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09*, pages 331–340. INSTICC Press, 2009.

[45] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168, 2006.

[46] Brian Tomasik, Phyo Thiha, and Douglas Turnbull. Tagging products using image classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 792–793, New York, NY, USA, 2009. ACM.

[47] N.Y. Khan, B. McCane, and G. Wyvill. Sift and surf performance evaluation against various image deformations on benchmark dataset. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 501–506, 2011.

[48] Johannes Bauer, Niko Sünderhauf, and Peter Protzel. Comparing several implementations of two recently published feature detectors. *University of Pennsylvania Law Review*, 154(3):477+, January 2006.

[49] Christoph H. Lampert, M.B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

[50] Han-Pang Chiu, L.P. Kaelbling, and T. Lozano-Perez. Virtual training for multi-view object class recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[51] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–272–7 vol.2, 2003.

[52] Jean Ponce, Svetlana Lazebnik, Fredrick Rothganger, and Cordelia Schmid. Toward True 3D Object Recognition. 2010.

[53] Min Sun, Hao Su, S. Savarese, and Li Fei-Fei. A multi-view probabilistic model for 3d object classes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1247–1254, 2009.

[54] Yu Xiang and S. Savarese. Estimating the aspect layout of object categories. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3410–3417, 2012.

[55] M. Aly, M. Munich, and P. Perona. Indexing in large scale image collections: Scaling properties and benchmark. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 418–425, 2011.