Universidad de Alcalá

*Campus Universitario*
*Dpto. de Teoría de la Señal y Comunicaciones*
*Ctra. Madrid-Barcelona, Km.* 36.6
*28805 Alcalá de Henares (Madrid)*
*Telf: +34 91 885 88 99*
*Fax: +34 91 885 66 99*

D. SATURNINO MALDONADO BASCÓN, Catedrático de Universidad del Área de Conocimiento de Teoría de la Señal y Comunicaciones de la Universidad de Alcalá,

CERTIFICA

Que la tesis titulada "**Estudio, diseño y optimización de algoritmos para la aplicación de técnicas de aprendizaje estadístico al procesado digital de imágenes**", presentada por D. Hilario Gómez Moreno, realizada en el Deparatamento de Teoría de la Señal y Comunicaciones bajo mi dirección, reune méritos suficientes para optar al grado de Doctor, por lo que puede procederse a su depósito y lectura.

Alcalá de Henares, 13 de diciembre de 2011.

Fdo: Dr. D. Saturnino Maldonado Bascón

Universidad de Alcalá

*Campus Universitario*
*Dpto. de Teoría de la Señal y Comunicaciones*
*Ctra. Madrid-Barcelona, Km.* 36.6
*28805 Alcalá de Henares (Madrid)*
*Telf: +34 91 885 88 99*
*Fax: +34 91 885 66 99*

D. Hilario Gómez Moreno ha realizado en el Departamento de Teoría de la Señal y Comunicaciones y bajo la dirección del Dr. D. Saturnino Maldonado Bascón, la tesis doctoral titulada "**Estudio, diseño y optimización de algoritmos para la aplicación de técnicas de aprendizaje estadístico al procesado digital de imágenes**", cumpliéndose todos los requisitos para la tramitación que conduce a su posterior lectura.

Alcalá de Henares, 13 de diciembre de 2011.

EL DIRECTOR DEL DEPARTAMENTO

Fdo: Dr. D. Saturnino Maldonado Bascón

Universidad de Alcalá

Escuela Politécnica Superior

Tesis Doctoral

# Estudio, diseño y optimización de algoritmos para la aplicación de técnicas de aprendizaje estadístico al procesado digital de imágenes

Autor: Hilario Gómez Moreno
Director: Dr. Saturnino Maldonado Bascón

13 de diciembre de 2011

*A mi mujer Elia, a mis hijas Lucía y Elsa y a mis padres por apoyarme. A mis compañeros por ayudarme.*

# Resumen

Esta tesis doctoral parte de la hipótesis de que ciertas herramientas de aprendizaje estadístico como las máquinas de vectores soporte (SVM) o las redes neuronales RBF pueden ser aplicadas en tareas de procesado de imagen de bajo nivel. Estas tareas de procesado son las primeras de la cadena en un esquema básico de procesamiento de imágenes. Las herramientas de procesado estadístico han sido usadas en tareas de alto nivel (reconocimiento de patrones) con excelentes resultados pero su potencial no ha sido estudiado completamente en tareas de bajo nivel.

Concretamente, en esta tesis se hace un estudio de las tareas de detección de bordes, segmentación de imágenes en color y eliminación de ruido impulsivo, tratando de definir alternativas a las técnicas ya existentes pero basadas en el uso de SVM. A lo largo de la misma se demuestra que las alternativas propuestas pueden igualar o mejorar las técnicas ya establecidas con la ventaja adicional de la flexibilidad y adaptación conseguida gracias a técnicas de aprendizaje basadas en imágenes sintéticas y a la facilidad de entrenamiento de las SVM.

Las tareas de procesado tratadas en esta tesis tienen en común la posibilidad de definir esquemas de clasificación y de regresión en los que se pueden aplicar los algoritmos de aprendizaje estadístico. Concretamente en la detección de bordes hay que clasificar los píxeles de la imagen entre aquellos que pertenecen a un borde y los que no, en la segmentación en color hay que clasificar los píxeles que pertenecen a un color y los que no y en la eliminación de ruido impulsivo es necesario detectar los píxeles que son ruidosos y para reconstruirlos puede utilizarse la regresión.

En la realización de esta tesis se han abordado, por tanto, las siguientes tareas correspondientes a cada una de las técnicas de procesado estudiadas:

- En la detección de bordes se ha realizado un estudio inicial del estado del arte para conocer los esquemas más comunes de detección, sus ventajas e inconvenientes. Posteriormente se ha definido un esquema de detección de bordes basado en la clasificación de los píxeles mediante SVM. En este esquema se ha dado bastante importancia a la definición del entrenamiento, que se realiza mediante imágenes creadas de manera sintética. Se ha realizado, así mismo, un estudio de los mejores parámetros de entrenamiento incluyendo el kernel de las SVM. Siguiendo el esquema de detección, se han definido técnicas para la obtención de una imagen de gradiente a partir de los datos de clasificación y, por último, se han definido esquemas de marcado de píxeles adaptando algunas técnicas existentes y proponiendo algunas soluciones novedosas. Por último, se han realizado pruebas de funcionamiento tanto de manera visual como con medidas objetivas, incluyendo algunas en las que el detector propuesto se aplica sobre imágenes con ruido añadido.

- En la segmentación en color, se han analizado distintas técnicas existentes para conocer los puntos donde se pueden mejorar. Tras este análisis, se ha propuesto la segmentación de los colores de la imagen con un esquema sencillo de detección en el espacio de color RGB. Este esquema de detección se ha comparado con otros ya existentes en una tarea concreta, el reconocimiento de señales de tráfico, demostrándose que la técnica propuesta puede ser una alternativa factible y que, mediante su uso con tablas de búsqueda, se puede mejorar el rendimiento y aplicarla en procesado de tiempo real.

- En el apartado de detección de ruido impulsivo se han estudiado varias técnicas existentes con distintos esquemas de aplicación. Entre ellos, se ha elegido el esquema de detección con reconstrucción posterior por los buenos resultados obtenidos. Para la detección se han definido imágenes de entrenamiento sintéticas y se han probado las SVM y las redes neuronales RBF como posibles clasificadores. Sus buenos resultados en esta tarea de clasificación y la facilidad de generación del entrenamiento permiten asegurar que esta es una buena alternativa a la detección mediante otras técnicas. En el apartado de reconstrucción se han probado filtros de mediana ya existentes, pero modificados para mejorar su funcionamiento, y esquemas de regresión con SVM, nuevamente basados en entrenamiento sintético. Los resultados de calidad obtenidos con distintas medidas permiten afirmar que las técnicas aquí propuestas son una alternativa a las existentes y, dependiendo de las imágenes y los niveles de ruido, pueden ofrecer mejores resultados.

# Abstract

This thesis starts on the hypothesis that certain statistical learning tools such as support vector machines (SVM) or RBF neural networks can be applied in low-level image processing tasks. These processing tasks are the first in a chain basic scheme of image processing. Statistical processing tools have been used in high-level tasks (pattern recognition) with excellent results but its potential has not been fully studied in low-level tasks.

Specifically, in this thesis a study of the tasks of edge detection, color images segmentation and impulsive noise removal is made, trying to define alternatives to existing techniques, but based on the use of SVM. Throughout it is shown that alternative proposals can meet or exceed the established techniques with the added advantage of flexibility and adaptation achieved by learning techniques based on synthetic images and easiness of SVM training.

While processing tasks discussed in this thesis may seem diverse, they have in common the ability to define classification and regression schemes in which you can apply statistical learning algorithms. Specifically, in the edge detection it is needed to classify the image pixels between those that belong to an edge and those outside, on the color segmentation it is needed to classify the pixels that belong to one color and those that do not and in impulsive noise removal it is necessary to detect the pixels that are noisy and can be reconstructed using the regression.

In this thesis have been addressed, therefore, the following tasks for each of the processing techniques studied:

- In the edge detection, it was performed an initial state of the art survey to understand the most common detection schemes, their advantages and disadvantages. Subsequently, there was defined an edge detection scheme based on pixel classification by SVM. In this scheme it has been given enough importance to the definition of training, which is done using synthetically generated images. It has made, also, a study of the best training parameters including kernel of SVM. Following the detection scheme, we have defined techniques for obtaining a gradient image from the classification data, and finally, we have defined schemes to mark pixels adapting some existing techniques and proposed some novel solutions. Finally, performance tests have been performed both visually and with objective measures, including some in which the proposed detector is applied to noisy images.

- In color segmentation, we have analyzed various existing techniques to understand the points where they can be improved. After this analysis, it is proposed to detect the colors of the image with a simple scheme of detection in the RGB color space. This detection scheme has been compared with existing ones on a particular

task, the recognition of traffic signs, demonstrating that the proposed technique can be a feasible alternative and that by its use with lookup tables the performance improvement allows real time processing.

- In the section on impulse noise detection, we have studied several existing techniques with different application schemas. Among them was chosen the detection scheme with subsequent reconstruction, due to good results. For detection, we have defined synthetic training images and tested the SVM and RBF neural networks as potential classifiers. Their success in classification task, and the ease of generation of training allowsn to ensure that this is a good alternative to other detection techniques. In the area of reconstruction, median filters have been tested, but modified to improve their performance, and regression SVM schemes, based on synthetic training again. Quality results obtained with different measures, allow us to state that the techniques proposed here are an alternative to existing ones and, depending on the images and noise levels, may provide better results.

# Glosario de abreviaturas y acrónimos

**AMF** Adpatative Median Filter

**CAD** Índice Cromático/Acromático

**CER** Eliminación de bordes en espacio de color

**CIE** Commission International de l'Eclariage

**CWM** Center Weighted Median

**DoG** Diferencia de Gaussianas

**DS-FIRE** Dual Step-Fuzzy Inference Ruled by Else-action

**ERBF** Exponential Radial Basis Function

**ERM** Minimización del riesgo empírico. Empirical Risk Minimization

**FIDRM** Fuzzy Impulse noise Detection and Reduction Method

**FOM** Pratt's Figure Of Merit

**FSB** Fuzzy Similiraty Based

**GER** Eliminación de bordes en escala de gris

**HSET** Umbralizado en HS mejorado.

**HSI** Espacio de Tono, Saturación e Intensidad. Hue, Saturation, Intensity

**HST** Umbralizado en HS (Tono y saturación)

**Lab y Luv** Espacios de color perceptivos

**LoG** Laplaciana de una gaussiana (Laplacian of a Gaussian)

**LUT** Tabla de búsqueda

**MMEM** Minimum-Maximum Exclusive Mean

**NTSC** Sistema de televisión en color americano. National Television System Committee

**OST** Umbralizado en el espacio Ohta

**PAL** Sistema de televisión en color europeo. Phase Alternating Line

**RBF** Funciones de base radial. Radial Basis Functions

**RGBNT** Umbralizado en RGB Normalizado

**RGB** Espacio Rojo, Verde, Azul. Red, Green, Blue

**rgb** Espacio de color RGB normalizado

**ROM** Rank-Ordered Mean

**SMVC** Detección de color con SVM

**SOB** Índice de sobresegmentación

**SRM** Minimización del riesgo estructural. Structural Risk Minimization

**SVM** Máquinas de Vectores Soporte. Support Vector Machines

**VC** Vapnik-Chervonenkis

**XYZ** Espacio de color XYZ

**YIQ** Espacio de color YIQ

**YUV** Espacio de color YUV

# Índice general

# Lista de figuras

# Lista de tablas

# Capítulo 1

# Introducción

## 1.1. Contexto de la investigación

Hoy en día son cada vez más los sistemas diseñados para la realización automática de ciertas actividades que, o son tediosas, o en las que un operador humano no sería eficiente. Estos sistemas se basan en algoritmos de inteligencia artificial que buscan la aplicación de principios estadísticos o biológicos a problemas complejos.

Esta tesis pretende la aplicación de estos mismos principios pero en tareas llamadas de "bajo nivel" en el procesado de imagen. Estas tareas son las iniciales de cualquier sistema en el que se realice algún procesado en el que se usen imágenes. Podrían ser, por ejemplo, la reducción del ruido que pueda tener la imagen registrada, la mejora de la iluminación de la escena, la segmentación de los objetos, la reducción del emborronamiento, etc. La parte final del sistema, en la que se aplican los algoritmos de más alto nivel, es sobre la que recae normalmente la atención pues es, en gran manera, la que decide el resultado. Sin embargo, está demostrado que el funcionamiento de estos procesos finales están claramente influidos por el procesamiento inicial.

## 1.2. Justificación y objetivos

Partiendo de la idea de que los algoritmos de bajo nivel se pueden también beneficiar del uso de técnicas de procesado estadístico, en esta tesis se definen maneras de aplicarlos a algunas tareas básicas de procesado de imagen.

En la ilustración de la figura 1.1 puede verse una justificación de la importancia de los procesos de bajo nivel, que a veces no se ven y parecen prescindibles, pero son la base del procesado posterior. Tomando esa ilustración como guía, y dentro de las "raíces" ahí definidas, se han tomado como objetivo la mejora o, por lo menos, dar un nuevo enfoque en las tareas de: mejora de la imagen (concretamente la reducción de ruido impulsivo), detección de bordes y segmentación de imágenes en color.

Para ello se ha buscado la aplicación de herramientas de clasificación o regresión basadas en el tratamiento estadístico de los datos (ampliamente utilizadas en tareas de alto nivel). Concretamente se ha hecho hincapié en la aplicación de las Máquinas de Vectores Soporte (Support Vector Machines) o métodos similares, como las redes neuronales RBF (Radial Basis Functions) en las tareas de bajo nivel.


Fuente: *Image Processing. The fundamentals*. Maria Petrou, 1999. Por cortesía de la autora

Figura 1.1: Importancia de las técnicas de procesado de bajo nivel. "*No me gustan las raíces. ¡Llenas de barro y otras cosas! ¡Pero los frutos si son agradables! ¡Cortaré las raíces y dejaré sólo la parte de arriba!*".

Durante la investigación se pretende optimizar el uso de dichas herramientas y conseguir algoritmos alternativos a los muchos ya existentes en las tareas de bajo nivel elegidas.

## 1.3. Organización de la memoria

La memoria está organizada en cinco capítulos de los que tres tienen una estructura similar.

El capítulo 2 es una introducción a los métodos de aprendizaje estadístico. En esa introducción se repasan conceptos de redes neuronales y SVM relativos a su aplicación en tareas de clasificación y regresión.

Los capítulos 3, 4 y 5 están organizados de manera similar, con su estudio sobre el estado del arte, presentación de algoritmos, presentación de resultados y un resumen final. En el capítulo 3 se realiza el estudio de la detección de bordes en imágenes en escala de grises, el capítulo 4 es el dedicado a la segmentación de imágenes en color y por último el 5

presenta la investigación sobre la eliminación de ruido impulsivo en imágenes. Todos tienen como hilo conductor la aplicación del aprendizaje estadístico aunque las características propias de cada tarea hacen que los caminos y procesos de aplicación sean distintos.

Por último, en el capítulo 6 se presentan la conclusiones generales, se resumen las principales aportaciones originales de esta tesis, y se presentan las líneas futuras de investigación.

Posteriormente se referencia la bibliografía utilizada en la realización de la tesis y finalmente se presentan las publicaciones en las que aparece parte del trabajo desarrollado en la misma. El documento termina con un índice alfabético que facilita la búsqueda de información.

# Capítulo 2

# Herramientas de aprendizaje estadístico

En este capítulo se presentan de manera breve algunas de las herramientas de aprendizaje estadístico que se han explorado y usado en la realización de esta tesis. Se presentan de una manera no exhaustiva pero dejando claro las bases de su funcionamiento, sus ventajas e inconvenientes y los parámetros que hay que definir en su uso. Dichas herramientas se han usado con dos fines distintos, por un lado como clasificadores y por otro para obtener funciones de regresión tanto lineales como no-lineales:

- Los clasificadores tienen como objetivo asignar a un determinado vector una clase o agruparlo dentro de un tipo determinado previamente establecido. Dicho vector caracterizará de alguna manera el problema de clasificación escogido, que puede ser desde el reconocimiento de objetos al diagnóstico automático de enfermedades. El campo de aplicación será todo aquel en el que haya que agrupar algún tipo de objeto o señal dentro de unas clases preestablecidas.

  El clasificador definirá un modelo para cada clase del problema en cuestión, de tal forma que la clase a la que pertenece un elemento se pueda calcular a partir de los datos que definen el elemento. Por lo tanto, el objetivo de un clasificador es asignar, de forma lo más precisa posible, una clase a nuevos elementos previamente no estudiados. Para ello, inicialmente deberemos obtener un conjunto de vectores cuya clase sea conocida (Conjunto de entrenamiento), a partir de la cual, mediante distintos procesos que dependerán del algoritmo utilizado, se "aprenderá" la forma de clasificación para que nuevos datos sean correctamente clasificados.

- El objetivo de la regresión es encontrar una función que relacione datos de entrada con valores de salida y que no tienen una relación matemática conocida o están afectados por el ruido. Dicha función será óptima en algún sentido estadístico (normalmente mínimos cuadrados) y ofrecerá la mejor curva de ajuste posible para los datos presentados.

  Los principios estadísticos en los que se basan los algoritmos de cálculo de funciones de regresión están íntimamente relacionados con los utilizados por los clasificadores. Por ello, generalmente, las herramientas que se usan en clasificación pueden usarse, con ligeras modificaciones, también en problemas de regresión.

## 2.1. Redes neuronales artificiales

Existen varias definiciones de lo que es una red neuronal, unas hacen más hincapié en la descripción funcional y otras en la analogía con el mundo biológico. A continuación se presenta la obtenida de [Haykin98] que define con bastante claridad las características de una red neuronal:

> *Una red neuronal es un procesador distribuido masivamente paralelo hecho de unidades simples de proceso, las cuáles tienen una natural inclinación para almacenar conocimiento experiencial y lo tienen disponible para su uso. Se parece al cerebro en dos aspectos:*
>
> 1. *El conocimiento se adquiere por la red, a partir de su entorno, mediante un proceso de aprendizaje.*
> 2. *Los pesos de la conexión entre neuronas, conocidos como pesos sinápticos, se usan para almacenar el conocimiento adquirido.*

Los primeros trabajos en redes neuronales artificiales surgieron con la introducción del modelo simplificado de neurona propuesto por McCulloch y Pitts en el año 1943 ([Mcculloch43]). Estos serían modelos de las neuronas biológicas y componentes para circuitos que podían realizar tareas de procesado. En el libro "Perceptrones" ([Minsky69]) de Minsky y Papert se mostraron las deficiencias en el modelo del perceptrón y la mayor parte de los grupos de investigación abandonaron el tema y buscaron otros modelos. Entre los que continuaron investigando sobre las redes neuronales artificiales se puede mencionar Teuvo Kohonen [Kohonen72, Kohonen76, Kohonen81], Stephen Grossberg [Grossberg76a, Grossberg76b, Grossberg75] o James Anderson [Anderson72b, Anderson72a] entre otros. El interés en las redes neuronales artificiales vuelve a emerger a principios de los años ochenta con nuevos avances teóricos (principalmente el algoritmo de la retro-propagación del error) y ayudados por las mejoras en los procesadores.

Las redes neuronales artificiales pueden ser descritas como "modelos computacionales" que pueden adaptarse o aprender, generalizar o agrupar u organizar datos, y que se caracterizan por usar el procesamiento paralelo. Algunas de estas propiedades las cumplen otros modelos y, por tanto, surge la pregunta de hasta dónde la aproximación basada en redes neuronales es mejor para una aplicación que otros modelos. Hasta la fecha no hay una respuesta clara a esta pregunta. En muchos libros, al hablar de redes neuronales se describe su paralelismo con modelos biológicos. Pero los modelos de redes neuronales artificiales no dejan de ser una "super-simplificación" de estos sistemas que son enormemente complejos. En este capítulo no se pretende describir exhaustivamente toda la teoría de las redes neuronales, sino ofrecer una idea de sus fundamentos y características. Por ello se presentan sólo algunos modelos de redes neuronales, dejando fuera otros modelos como los mapas auto-organizativos de Kohonen que tienen una estructura algo diferente.

Una red neuronal artificial está constituida por un conjunto de elementos de procesado simple llamadas neuronas. Una neurona es un dispositivo sencillo (Figura 2.1) formado por una serie de entradas y una salida. Cada neurona acepta como entrada las salidas procedentes de otras neuronas, siendo la entrada efectiva a la neurona la suma ponderada de las entradas reales a dicha neurona. La salida de la neurona dependerá de la función de activación que tenga programada internamente y que normalmente es no lineal. La suma

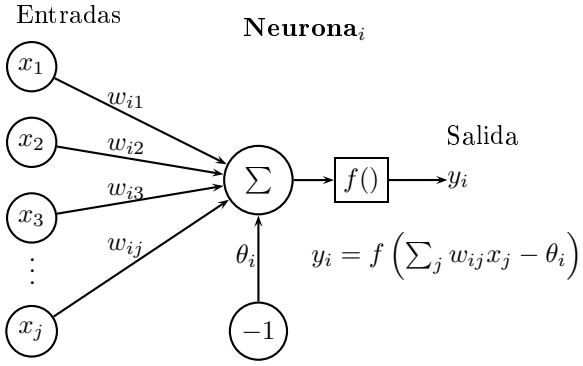



Figura 2.1: Ejemplo de neurona artificial.

de las entradas ponderadas pasarán a dicha función que dará un valor de salida acorde. Cada neurona realiza una tarea sencilla: recibe la información de entrada de los vecinos o del exterior y la usa para calcular una señal de salida que se propaga a otras unidades.

En general, las neuronas se organizan en capas (Figura 2.2). Dependiendo de su función en la red, se distinguen tres tipos de neuronas o capas:

1. Las neuronas cuya activación son los datos de entrada del problema y que forman la capa de entrada. El número de neuronas en la capa de entrada depende de la dimensión del problema a tratar.

2. Las neuronas que dan lugar a la salida de la red forman la capa de salida. El número de neuronas de la capa de salida depende del número de clases que queramos diferenciar. Normalmente se dispone de tantas neuronas como elementos se quiera diferenciar.

3. El resto de neuronas se agrupan en una serie de capas sin conexión directa ni con la entrada ni con la salida y es lo que se conoce como capas ocultas. Para las capas intermedias no existe un criterio para determinar ni el número de capas ni el de neuronas por capa, debe ser ajustado para el problema en cuestión.

El papel fundamental de las redes neuronales artificiales está en su uso como clasificadores o como funciones de regresión que aprenden de los datos de entrenamiento y pueden inferir correctamente las mejores funciones de decisión.

### 2.1.1. Perceptrón Multicapa.

Dentro de las redes neuronales artificiales una de las más conocidas y usadas es el perceptrón multicapa. Este tipo de red está constituido por un conjunto de neuronas que se comunican enviándose señales entre sí a través de un conjunto de conexiones ponderadas, siendo el caso más sencillo de red neuronal. Se corresponde con la estructura presentada en la figura 2.2. Es una red en la que la información fluye directamente de la entrada a la salida sin ninguna realimentación y son conocidas como de propagación directa. Aunque la estructura puede permitir el uso de varias capas ocultas, normalmente sólo se usa una para simplificar el entrenamiento y evitar problemas de sobreentrenamiento.

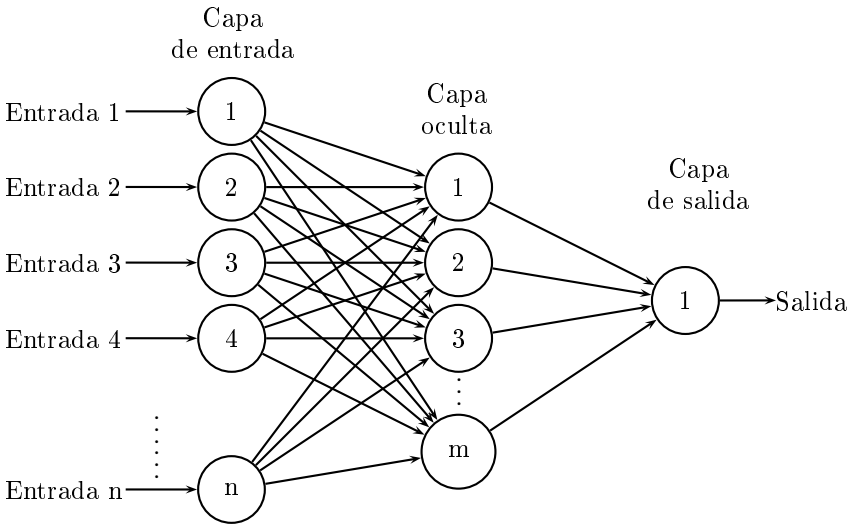



Figura 2.2: Ejemplo de red neuronal.

La primera tarea que debe realizarse con una red neuronal es el ajuste de los pesos entre neuronas, ya que inicialmente se les da un valor aleatorio. Generalmente se usa un sistema de entrenamiento de tipo supervisado; durante la fase de entrenamiento se le presentan a la red tanto los patrones que queremos que diferencie como los resultados que debería proporcionar cada patrón. La red va reajustando los pesos de las conexiones en función del error cometido, hasta que para los patrones iniciales, la salida de la red neuronal es la correcta. Una vez realizada esta fase, los pesos no varían y la red está lista para clasificar correctamente los nuevos datos que se le presenten. Una vez entrenada, la salida de una neurona $i$ vendrá dada por la expresión:

$$f_i(\mathbf{x}) = f_o\left[\sum_{j=1}^{N} \mathbf{w}_{oj}^i \left(f_h\left(\left\langle \mathbf{w}_h^j, \mathbf{x}\right\rangle + b_{hj}\right)\right)\right] + b_i, \tag{2.1}$$

donde $f_h(\ )$, $f_o(\ )$ son las funciones de activación de la capa oculta y de la capa de salida respectivamente, $\mathbf{w}_h^j$ es el vector de pesos de la neurona $j$ de la capa oculta, $\mathbf{w}_o^i$ es el vector de pesos de la neurona de salida, $N$ es el número de neuronas de la capa oculta y $b_{hj}$ y $b_i$ son las constantes de desplazamiento de cada neurona.

Para una discusión en detalle de los algoritmos, se recomienda la consulta de Bishop [Bishop96] y Haykin [Haykin98].

### 2.1.2. Funciones de base radial (RBF).

Las funciones de base radial (Radial Basis Functions o RBF) se introdujeron en [Broomhead88] como método alternativo al perceptrón multicapa para hacer ajuste de funciones no lineales. Posteriormente se añadieron métodos de entrenamiento rápido en [Moody89].

Funcionalmente, las RBF son redes neuronales de una capa oculta alimentados hacia delante con funciones de transferencia lineales en la salida y no lineales en la capa oculta (Figura 2.3). Constan de dos capas de redes de aprendizaje híbrido, similares al perceptrón multicapa en términos de estructura y activación en su capa supervisada, desde los nodos

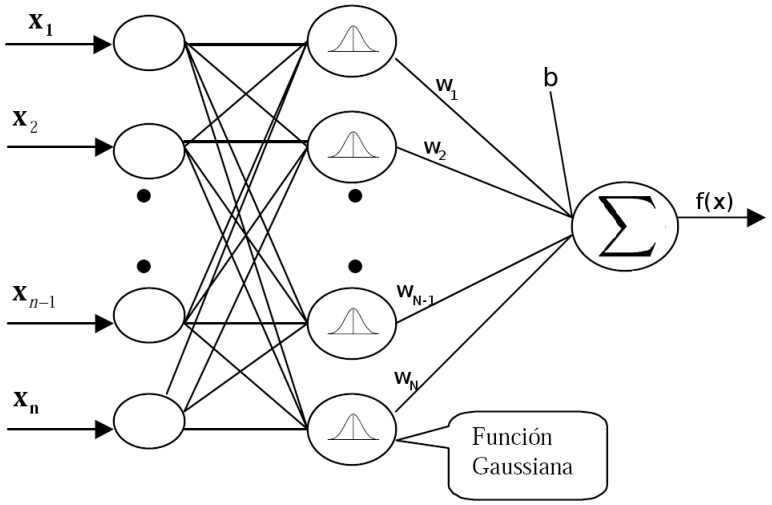


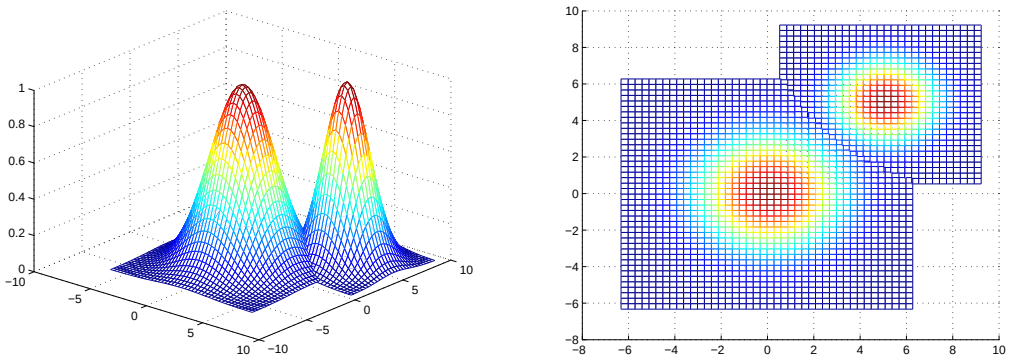

Figura 2.3: Ejemplo de red RBF.

ocultos a los nodos de salida. Sin embargo, las capas sin supervisar, desde la capa de entrada a la oculta, difieren en que en ellas existen funciones radiales individuales del tipo campana de Gauss para cada nodo oculto. Se necesita definir una distancia, que puede ser la distancia euclídea, entre el vector de entrada n-dimensional y el centro de la neurona oculta. El valor de salida aumentará cuando ambos valores estén cerca (de forma similar a cómo varía una curva gaussiana centrada en la neurona oculta). La forma general de la red radial será entonces:

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|) + b, \tag{2.2}$$

donde $N$ es el número de neuronas de la capa oculta, $w_i$ el peso de la salida de la neurona $i$ en la neurona de salida y $b$ es una constante de desplazamiento de la neurona de salida. La salida de la red será la suma ponderada de las respuestas de cada neurona oculta ($\phi$) y cada una de éstas tiene asignada una posición y una "anchura" (Figura 2.4) que define la naturaleza y ámbito de la respuesta de la neurona, y es equivalente a la desviación estándar de la anchura de la campana de Gauss. De tal forma que valores altos indican que hay muchos puntos que van a ser considerados. Esto significa que en contra de lo que sucede en el perceptrón multicapa, las RBF poseen una función de activación que relaciona la proximidad relativa entre el conjunto de prueba y el de entrenamiento. Ello permite una medida directa de la confianza en la salida de la red para un patrón dado. Si un patrón es muy diferente de aquellos con los que la red ha sido entrenada, la respuesta será muy baja o nula.

Comparadas con los perceptrones multicapa las RBF producen aproximaciones locales a los datos, manejan mejor un número grande de datos y se entrenan más rápido, sin embargo, necesitan un mayor número de neuronas para un mismo problema. En todo caso, ambas pueden utilizarse en problemas de clasificación y como funciones de aproximación ya que se puede demostrar que son aproximadores universales.



**(a)** ***Ejemplo de Gaussianas bidimensionales*** **(b)** ***Vista superior de ambas gaussianas donde se aprecia la zona de efecto de cada una***

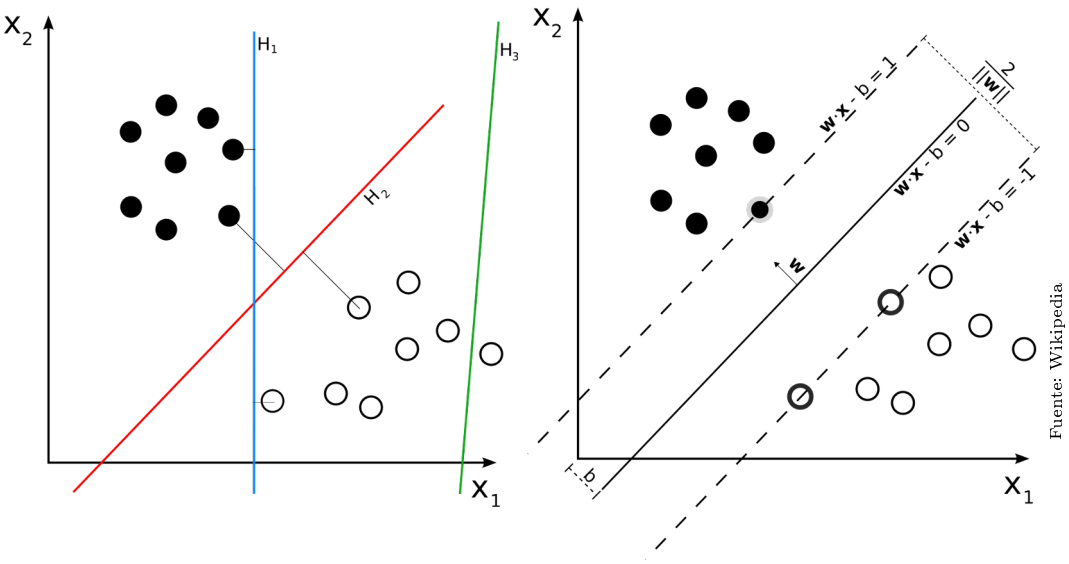
Figura 2.4: Ejemplo de gaussianas no normalizadas.

## 2.2. Máquinas de vectores soporte. (SVM)

El propósito de este apartado consiste en introducir las ideas básicas que fundamentan las máquinas de vectores soporte utilizadas en la presente tesis. Se han usado en multitud de aplicaciones de reconocimiento de patrones, como reconocimiento de escritura a mano [Cortes95], reconocimiento de objetos en 3D [Blanz96], categorización de texto [Joachims02], búsqueda en bases de datos de imágenes [Tao06], clasificación de texturas [Kim02], etc. La lista de aplicaciones no hace más que crecer debido a las buenas prestaciones obtenidas por las SVM. Aunque son menos las aplicaciones en procesado de bajo nivel como las presentadas en esta tesis, muestran un incremento notable en los últimos años como puede verse en [Zheng04, Zou05, Pankajakshan07, Wang07, Bravo07].

Las máquinas de vectores soporte fueron desarrolladas por Vapnik ([Boser92, Cortes95, Vapnik95, Vapnik98]) y tienen su base teórica en la teoría estadística sobre la Minimización del Riesgo Estructural (SRM) frente a la minimización del riesgo empírico (ERM) propio de las redes neuronales. Al igual que los perceptrones multicapa o las redes de funciones de base radial se pueden usar tanto para clasificación como para regresión. En poco tiempo se han convertido en una herramienta de uso generalizado en el campo de reconocimiento de patrones debido a los buenos resultados obtenidos. La aplicación más sencilla de esta técnica es el problema de clasificación binaria (sólo hay definidas dos clases) aunque se puede ampliar a la multiclasificación.

La idea subyacente consiste en encontrar una función de clasificación dentro de un conjunto de posibles clasificadores que minimice la probabilidad de error empírico (error cometido al recibir nuevos datos) aunque manteniendo la complejidad del clasificador en unos límites aceptables. Dicha complejidad está relacionada con la conocida como dimensión VC (Vapnik-Chervonenkis [Vapnik71]) que es pieza central en la teoría de aprendizaje estadístico. La combinación de minimización de riesgo empírico junto con minimización de riesgo estructural permite encontrar clasificadores que generalicen de manera adecuada sin llegar al sobreentrenamiento.




(a) (b)

Figura 2.5: Problema de separación óptima de patrones en dos dimensiones. Datos separables linealmente.

### 2.2.1. Clasificación mediante SVM.

La idea básica de la clasificación con SVM es encontrar la mejor función de separación entre dos clases determinadas distribuidas en un espacio n-dimensional. En la figura 2.5(a) se muestra un ejemplo de separación entre clases en el que aparecen dibujadas varias fronteras de separación. La frontera verde ($H_3$) claramente no separa las clases pero cualquiera de las otras dos sí lo hace. De hecho se podrían trazar infinitas fronteras de separación lineales entre ambas clases. ¿Cuál sería entonces la frontera óptima de separación? En [Vapnik98] se prueba que minimizar la combinación de los riesgos empírico y estructural es equivalente a encontrar el hiperplano que se encuentra a la distancia máxima de las muestras de entrenamiento más cercanas para las dos clases y que esa sería la frontera de decisión óptima. Esto sólo sería posible en el caso de que la función de separación fuera lineal. En la figura 2.5(b) aparece dicha frontera de decisión óptima que es la más alejada de las dos clases (la distancia será igual a $\frac{2}{\|\mathbf{w}\|}$). En dicha figura aparecen también marcados los vectores que están más cercanos a la frontera de decisión y que serían los conocidos como vectores soporte pues son los que formarán parte de la función de decisión. El resto de vectores no aportan información para dicha función.

En el caso de una frontera de separación no lineal la clave del procedimiento consiste en establecer una correspondencia entre las muestras en el espacio de entrada y otro conjunto de vectores transformados en un espacio de dimensión mayor o igual, el llamado espacio de características, en el que la separación se puede hacer de manera lineal. Para realizar la transformación se utiliza una correspondencia previamente definida, llamada kernel. Es en este espacio de características en el que se construye el hiperplano óptimo que separa las dos clases. Por lo tanto, el procedimiento termina con una función de decisión lineal en el espacio de características pero que no lo es en el espacio inicial.

**Datos separables linealmente.** Para el caso lineal, el problema se puede expresar de forma matemática partiendo de un set de datos de entrenamiento $\mathcal{E}$, con $n$ puntos de la forma $\mathcal{E} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\, y_i \in \{-1, 1\}\}_{i=1}^{n}$. La etiqueta $y_i$ puede ser $+1$ o $-1$, indicando la clase a la que pertenece el vector $\mathbf{x}_i$ que es real y $p$-dimensional.

Se quiere obtener el hiperplano con margen máximo que divide los vectores con $y_i = +1$ de aquellos que tienen $y_i = -1$. Cualquier hiperplano puede ser escrito sabiendo que los vectores $\mathbf{x}$ que pertenezcan a él satisfacen:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0. \tag{2.3}$$

El vector $\mathbf{w}$ es un vector normal al hiperplano definido. El parámetro $b$ es el offset, que determina cual de los posibles hiperplanos paralelos definidos por $\mathbf{w}$ se está usando.

Hay que buscar los valores de $\mathbf{w}$ y $b$ que hagan máximo el margen entre clases. Ese margen será la distancia entre los dos hiperplanos paralelos más alejados entre sí y que siguen separando los datos. Esos dos hiperplanos vendrán dados por:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 1, \tag{2.4}$$

y

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = -1. \tag{2.5}$$

En la figura 2.5(b) aparecen gráficamente dichos hiperplanos en el caso bidimensional y para clases linealmente separables. Geométricamente se puede llegar a que la distancia entre ambos será $\frac{2}{\|\mathbf{w}\|}$, por tanto, para maximizarla habrá que minimizar $\|\mathbf{w}\|$. Como no puede haber vectores pertenecientes al conjunto de entrenamiento entre ambos hiperplanos se tiene que cumplir para cada vector $\mathbf{x}_i$ o bien que

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 \qquad \text{para } \mathbf{x}_i \text{ de la primera clase,} \tag{2.6}$$

o

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 \qquad \text{para } \mathbf{x}_i \text{ de la segunda clase.} \tag{2.7}$$

Uniendo las dos expresiones se llega a:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \quad \text{para todo } 1 \leq i \leq n. \tag{2.8}$$

El problema de optimización será por tanto:

$$\begin{aligned} &\min_{\mathbf{w}} \quad \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle \\ &\text{sujeto a:} \\ &\qquad y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) \geq 1. \end{aligned} \tag{2.9}$$

La resolución de este problema nos lleva al uso de la teoría de minimización con restricciones que aplicada a este problema sugiere el uso de multiplicadores de Lagrange. La función de optimización que quedaría para este caso sería:

$$L\,(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) - 1] + \sum_{i=1}^{n} \alpha_i. \tag{2.10}$$

Esta es la que se conoce como forma primaria del problema. Sin embargo, este tipo de problemas se resuelve más fácilmente usando la forma dual del mismo a partir de la derivación parcial de la forma primaria y su igualación a 0. El nuevo problema de optimización sería entonces:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \tilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ &\text{sujeto a:} \\ &\quad \alpha_i \geq 0 \\ &\quad \mathbf{y}^T \boldsymbol{\alpha} = 0 \\ &\quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \\ &\quad \alpha_i [y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) - 1] = 0. \end{aligned} \tag{2.11}$$

La solución obtenida sería el hiperplano de máximo margen buscado que tendrá la forma:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}, \tag{2.12}$$

y en ella los valores de sólo algunas valores de $\alpha_i$ son distintos de 0 porque se tienen que cumplir las condiciones establecidas en el problema dual (ecuación 2.11) de forma que:

$$\begin{aligned} &\alpha_i = 0, \quad \text{si} \quad y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > 1 \\ &\alpha_i > 0, \quad \text{si} \quad y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) = 1. \end{aligned} \tag{2.13}$$

Por tanto, la expresión del hiperplano sólo dependería de los vectores cuyo $\alpha_i$ sea distinto de 0 que son los que caen exactamente en el margen y la expresión de la ecuación (2.12) sería:

$$\mathbf{w} = \sum_{i=1}^{N_{SV}} \alpha_i y_i \mathbf{x_i}, \tag{2.14}$$

donde $N_{SV}$ es el número de vectores soporte para el hiperplano de decisión. Es decir, de los vectores de entrenamiento, sólo serían necesarios para encontrar la solución aquellos más próximos a la frontera de decisión como ya se vio gráficamente en la figura 2.5(b).

Para obtener el parámetro $b$ simplemente hay que usar uno de los vectores soporte que cumplirá:

$$y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle - b) = 1, \tag{2.15}$$

y despejando quedaría:

$$b = \langle \mathbf{w}, \mathbf{x_i} \rangle - y_i. \tag{2.16}$$

Este valor sería válido para cualquier vector soporte elegido aunque para dar más robustez al cálculo se puede hacer la media de los valores obtenidos para todos los vectores soporte:

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\langle \mathbf{w}, \mathbf{x_i} \rangle - y_i). \tag{2.17}$$

Así, una vez realizada la minimización y obtenidos los valores de $\alpha_i$ y $b$ podemos formar la función de clasificación correspondiente para nuevos vectores $\mathbf{x}$ que será:

$$f(\mathbf{x}) = \text{signo}\left(\sum_{i=1}^{N_{SV}} \alpha_i y_i \langle \mathbf{x_i}, \mathbf{x}\rangle + b\right). \tag{2.18}$$

El procedimiento y los resultados obtenidos hasta ahora pueden usarse sólo directamente en el caso de que los vectores sean separables y que, además, esta separación sea mediante una frontera lineal. Sin embargo, en los casos más comunes de problemas de clasificación no se dan esas condiciones. Por tanto, el algoritmo debe ser ampliado para obtener una frontera de decisión en el caso de que los vectores no sean directamente separables y también para el caso en que la frontera de separación óptima no sea lineal.

**Datos no separables.** Hay problemas de clasificación en los que hay datos aislados pertenecientes a una clase mezclados con los de la otra clase a separar, es lo que se conoce como *outliers*. Esta situación se muestra gráficamente en dos dimensiones en la figura 2.6(a) donde los dos conjuntos de datos no son separables por una frontera lineal puesto que ambos conjuntos están mezclados.

Para resolver este tipo de problemas se pueden introducir algún tipo de relajación en las condiciones impuestas en el problema de minimización. Esto se puede hacer introduciendo unas variables positivas de pérdidas $\xi_i$ en las condiciones de la ecuación (2.9) ([Cortes95]) de forma que queden:

$$y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle - b) \geq 1 - \xi_i. \tag{2.19}$$

Cuando se produzca un error de clasificación $\xi_i$ excederá la unidad y, por tanto, la suma de dichas variables $\sum_i \xi_i$ será un límite superior para el número de errores cometido. Si se quiere que ese número de errores sea lo más pequeño posible se puede añadir un factor a la función de coste, de manera que el resultado final sea:

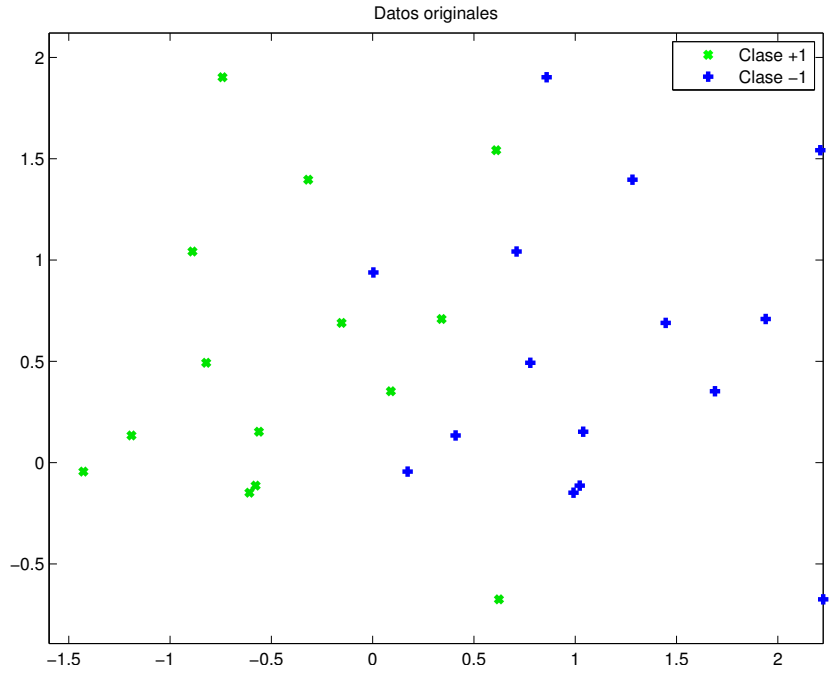
$$\begin{aligned} &\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{1}{2}\langle \mathbf{w}, \mathbf{w}\rangle + C\sum_{i=1}^{n} \xi_i \\ &\text{sujeto a:} \\ &\qquad y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \geq 1 - \xi_i \quad \xi_i \geq 0, \end{aligned} \tag{2.20}$$

donde $C$ es una constante de regularización (permite elegir un compromiso entre correcta clasificación y generalización) que debe ser elegida a priori por el usuario. A mayor valor de $C$ más importancia se le dará a los errores cometidos en la clasificación.

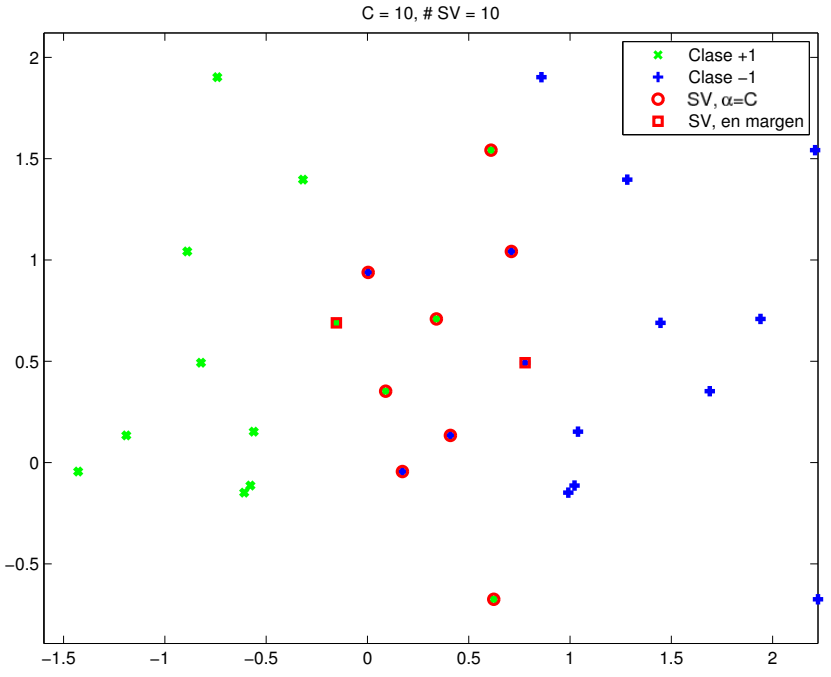
La forma de encontrar la solución a este problema es análoga a la presentada en el caso separable y queda reflejada en el ecuación (2.14) aunque en este caso los valores de los $\alpha_i$ tendrán un límite superior en $C$. La solución del vector $\boldsymbol{\alpha}$ clasifica los patrones de entrada en tres tipos diferentes:

1. Aquellos con $\alpha_i = 0$. Estos patrones son correctamente clasificados y no aportan información para la obtención del hiperplano de separación.

2. Si $0 < \alpha_i < C$ los patrones correspondientes están exactamente en los márgenes máximos. Estos sí aportan información y son conocidos como vectores soporte de tipo 1.




**(a)**




**(b)**

Figura 2.6: Ejemplo de datos no separables. (a) Datos iniciales, (b) Vectores soporte marcados con los dos tipos diferenciados.

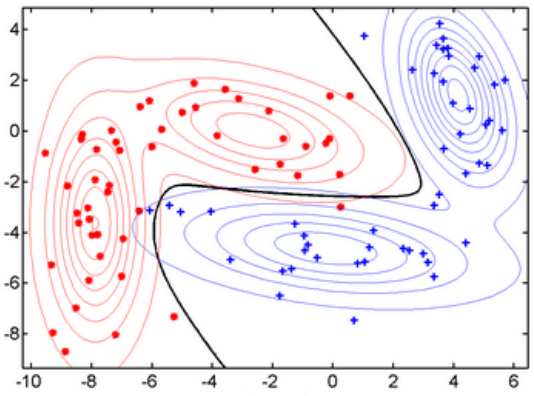



Figura 2.7: Ejemplo de datos no separables linealmente.

3. Con $\alpha_i = C$ los patrones correspondientes o bien están mal clasificados o caen dentro del margen. También son vectores soporte y se les conoce como de tipo 2.

Todos los vectores con $\alpha_i \neq 0$ formarán parte de la frontera de decisión que tendrá la misma forma que la presentada en la ecuación (2.18).

**Datos no separables linealmente.** El caso de datos no separables linealmente es muy común en problemas de clasificación (Figura 2.7). Para estos casos la teoría de las SVM lineales se extiende sin más que suponer que si existe una transformación que pase los datos a un espacio (espacio de características) en el que sí sean separables linealmente sólo habrá que encontrar el hiperplano de separación en ese nuevo espacio y luego volver al espacio inicial (Figura 2.8). Esta transformación no se conoce explícitamente sino que se utiliza a través del uso de kernel. Dichos kernel serán el resultado del producto escalar de la función de transformación necesaria,

$$\begin{aligned} k(\mathbf{x},\mathbf{y}) &= \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle \\ k(\mathbf{x},\mathbf{y}) &= k(\mathbf{y},\mathbf{x}) \rightarrow \mathbb{R}. \end{aligned} \tag{2.21}$$

La expresión anterior es conocida como el truco del kernel ("Kernel trick") ya que no es necesario conocer la transformación explícitamente puesto que para la obtención de la frontera de decisión sólo es necesario el producto escalar. Entonces la función de clasificación de la ecuación (2.18), en el caso de necesitar kernel sería:

$$f(\mathbf{x}) = \text{signo}\left(\sum_{i=1}^{N_{SV}} \alpha_i y_i k(\mathbf{x_i}, \mathbf{x}) + b\right). \tag{2.22}$$

Algunos de los kernel más habituales son:

- Kernel lineal: $k(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y} \rangle + c$, donde $c$ es una constante.
- Kernel polinomial: $k(\mathbf{x},\mathbf{y}) = (\alpha\langle \mathbf{x},\mathbf{y} \rangle + c)^d$ donde $\alpha$ y $c$ son constantes y $d$ es un entero positivo.

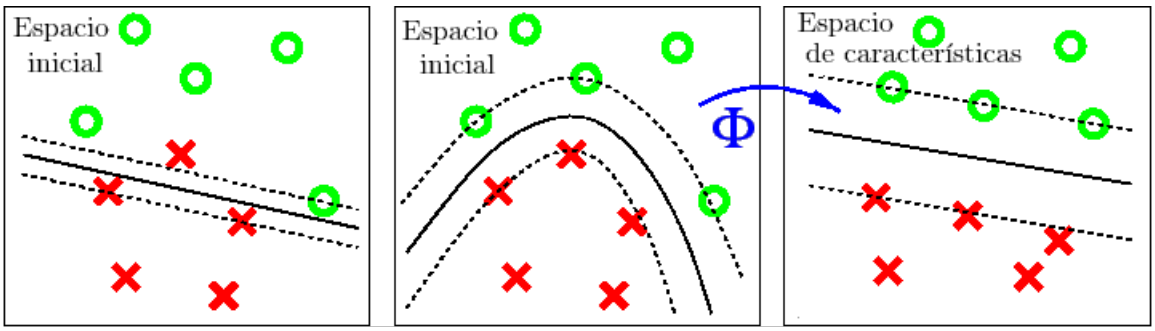



Figura 2.8: Transformación de datos usando el truco del kernel. (a) En el espacio inicial los datos no se pueden separar linealmente. (b) La frontera óptima no es lineal. (c) La frontera de decisión no lineal se corresponde con una lineal en el espacio de características. La transformación de espacios se hace mediante la función $\Phi$ que genera el kernel $k$.

- Kernel Gaussiano: $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma\|\mathbf{x} - \mathbf{y}\|^2\right)$. La constante $\gamma$ debe ser cuidadosamente fijada porque influye en el grado de generalización obtenido por la SVM que lo usa. Controla la anchura de la curva gaussiana a la que da lugar.

- Kernel ERBF (Exponential Radial Basis Function): $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma\|\mathbf{x} - \mathbf{y}\|\right)$. Está muy relacionado con el kernel gaussiano, la diferencia está en que la distancia no está elevada al cuadrado.

Existen un número importante de kernel desarrollados hasta el momento y que poseen características propias para determinados problemas. Una buena recopilación puede encontrarse en [Souza10].

### 2.2.2. Regresión mediante SVM.

Las máquinas de vectores soporte se pueden aplicar no solo al caso de la clasificación sino también a la regresión. En este caso se parte de unos datos de entrada formados por un conjunto de vectores asociados no a etiquetas que los clasifiquen sino a valores reales que posiblemente están afectados por ruido. El objetivo entonces es encontrar una función aproximada que ligue esos valores reales con los vectores de entrada. Dicha función será lo más próxima posible a los valores ya conocidos y además permitirá obtener valores nuevos no conocidos. El uso de SVM para regresión es especialmente adecuado en casos en que el número de datos es reducido, debido a su capacidad de generalización.

Cualquier algoritmo de regresión debe partir de una función de pérdidas con la que se mida el error cometido en la aproximación. En el caso de la clasificación el error es el número de vectores mal clasificados pero en el caso de la regresión no hay mala clasificación sino un mayor o menor error que se puede medir simplemente como una distancia entre el valor real y el valor estimado. Hay muchas funciones de pérdidas definidas en la literatura: lineales, cuadráticas, exponenciales, etc. La usada en la regresión con SVM fue presentada

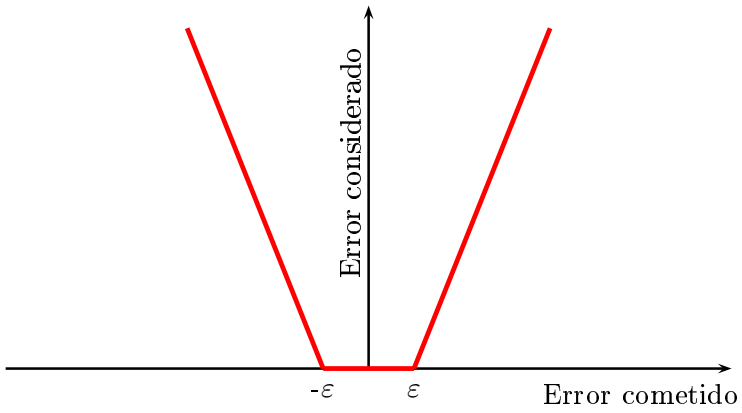



Figura 2.9: Función de pérdidas usada para la regresión con SVM. "$\varepsilon$-insensitive loss function".

por Vapnik y aparece definida gráficamente en la figura 2.9. Esa gráfica se corresponde con la expresión,

$$\mathcal{L}\left(y, f\left(\mathbf{x}\right)\right) = \begin{cases} 0 & \text{si } \left|y - f\left(\mathbf{x}\right)\right| \leq \varepsilon \\ \left|y - f\left(\mathbf{x}\right)\right| - \varepsilon & \text{resto} \end{cases}. \tag{2.23}$$

En esta función $\varepsilon > 0$ es una constante predefinida que controla la tolerancia al posible ruido presente en los datos. Puede verse que si el error cometido está por debajo de $\varepsilon$ no se tendrá en cuenta y si es mayor se tomará de forma lineal. Con esta función, la solución que se obtenga usando las SVM será dispersa ([Vapnik98]) mientras que cualquier otra no garantiza dicha dispersión.

**Regresión lineal.** Matemáticamente el problema que se plantea es que partiendo de un conjunto de datos:

$$\mathcal{D} = \left\{\left(\mathbf{x}_1, y_1\right), \left(\mathbf{x}_2, y_2\right), \cdots \left(\mathbf{x}_l, y_l\right)\right\}, \qquad x \in \mathbb{R}^n, y \in \mathbb{R}, \tag{2.24}$$

trataremos de aproximarlos con una función lineal,

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b. \tag{2.25}$$

La función de regresión óptima vendrá dada por el problema de optimización,

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}} \quad & \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{n} \left(\xi_i^+ - \xi_i^+\right), \\ \text{sujeto a} \quad & \begin{cases} y_i - \left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) & \leq \varepsilon + \xi_i^+ \\ \left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) - y_i & \leq \varepsilon + \xi_i^- \\ \xi_i^+, \xi_i^- & \geq 0 \end{cases}, \end{aligned} \tag{2.26}$$

donde $C$ debe ser fijado a priori y $\xi_i^-$, $\xi_i^+$ son variables de pérdidas que representan errores superiores e inferiores en las salidas de la función de estimación. En la figura 2.10 aparece una representación gráfica del significado de los parámetros expuestos hasta ahora. En la

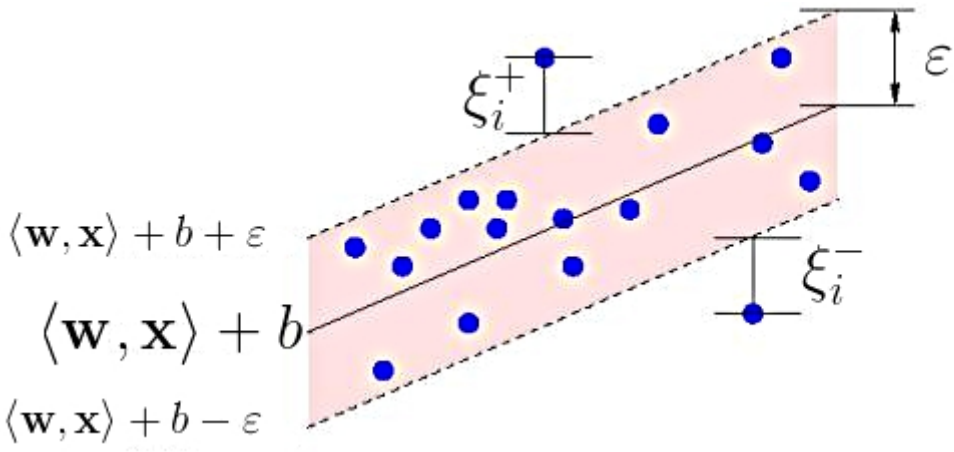



Figura 2.10: Regresión lineal.

ecuación (2.26) se ve que el problema presentado intenta buscar la función lineal lo más plana posible ("flat" en inglés) y que a la vez obtenga el menor error posible. El parámetro $C > 0$ será el encargado de dar más peso a una parte o a otra. Como se puede comprobar el problema es muy similar al presentado para la clasificación lineal y la solución se obtiene usando técnicas muy similares que implican el uso de multiplicadores de Lagrange y teoría de minimización. Quedará como resultado final la expresión,

$$f(\mathbf{x}) = \sum_{i=1}^{N_{SV}} (\alpha_i - \alpha_i^*) \langle \mathbf{x}_i, \mathbf{x} \rangle + b, \tag{2.27}$$

donde $\alpha_i$, $\alpha_i^*$ son los multiplicadores de Lagrange. Cumplen que $\alpha_i \alpha_i^* = 0$ y, por tanto, no pueden ser simultáneamente distintos de 0. Cuando alguno de los dos es distinto de 0 su vector asociado contribuye a la solución y es considerado un vector soporte.

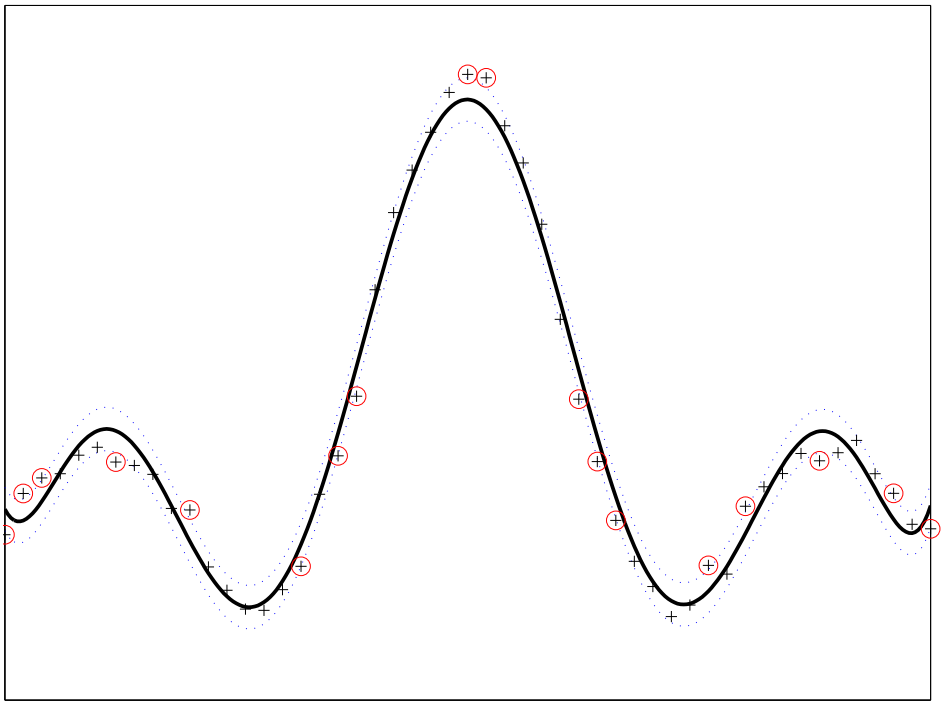
El valor de la constante $b$ se obtendrá sin más que buscar dos vectores soporte y usando,

$$b = -\frac{1}{2} \left( \langle \mathbf{w}, \mathbf{x}_r \rangle + \langle \mathbf{w}, \mathbf{x}_s \rangle \right). \tag{2.28}$$

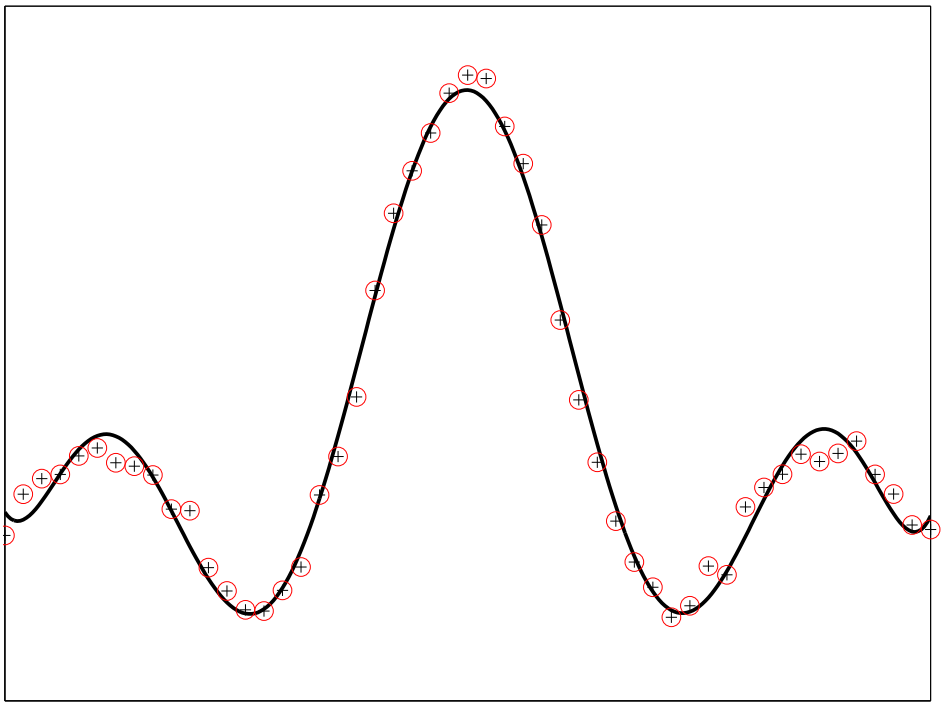
**Regresión no lineal.** Para el caso de la regresión no lineal (el más común) la idea es similar al caso de la clasificación no lineal, en el que los vectores eran transformados a un espacio en el que sí se pueden separar linealmente para luego transformar la frontera de decisión al espacio original. El uso de kernel hace posible esta transformación sin tener que realizarla explícitamente. Para la regresión se van a usar nuevamente kernel que permitan obtener la función de regresión en el espacio transformado. De esta manera la función de regresión a obtener será:

$$f(\mathbf{x}) = \sum_{i=1}^{N_{SV}} (\alpha_i - \alpha_i^*) k(\mathbf{x}_i, \mathbf{x}) + b, \tag{2.29}$$

y el valor de $b$ se obtendría de manera similar a la presentada en la ecuación (2.28).



**(a)** ***Kernel Gausiano RBF***, $\gamma = 1$, $C \to \infty$, $\varepsilon = 0.05$. $N_{SV} = 18(35.3\,\%)$.



**(b)** ***Kernel Gausiano RBF***, $\gamma = 1$, $C \to \infty$, $\varepsilon = 0$. $N_{SV} = 51(100\,\%)$.

Figura 2.11: Ejemplos de regresión no lineal.

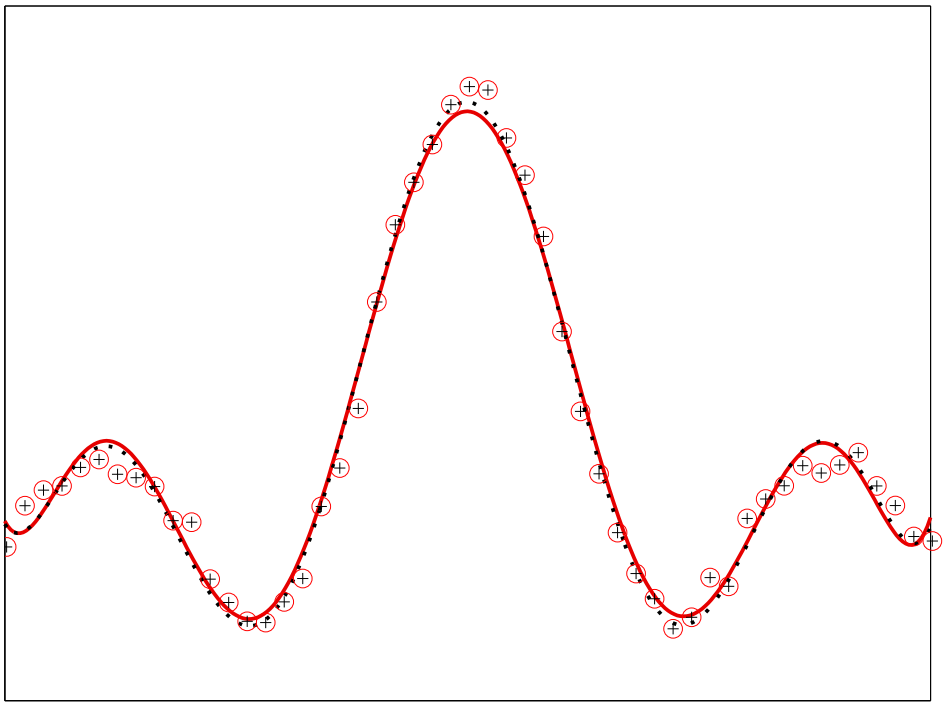


Figura 2.12: Comparación de los ejemplos de la figura 2.11. En rojo continuo aparece el caso de $\varepsilon = 0.05$ y en negro discontinuo el caso de $\varepsilon = 0$.

En la figura 2.11 se presentan dos ejemplos de aplicación de la regresión en 2 dimensiones. Se ha partido de una función $sinc(x)$ de la que se han obtenido 51 muestras equiespaciadas y con un pequeño ruido uniforme añadido.

En la figura 2.11(a) se ha usado un kernel gaussiano con $\gamma = 1$, la constante $C$ elegida es $\infty$ dando el peso total a los errores cometidos y la constante de insensibilidad $\varepsilon = 0.05$. Con estos parámetros la solución obtenida tiene sólo 18 vectores soporte (círculos rojos) de los 51 posibles (cruces) indicando la dispersión inherente a las SVM. Puede observarse que la función obtenida es muy similar a la original aunque los datos de partida tenían ruido añadido por lo que la planicidad ("flatness") también se ha conseguido. En color azul y discontinuo aparece marcada la zona de insensibilidad fijada por el valor de $\varepsilon$.

En la figura 2.11(b) se presenta un caso similar al anterior pero en el que $\varepsilon = 0$ y, por tanto, todos los errores son considerados en la función de regresión. El resultado es que todos los datos de entrenamiento se convierten en vectores soporte y se pierde la dispersión de la solución. Sin embargo, la solución es muy similar al caso anterior, como puede observarse en la figura 2.12. Por tanto, aunque de manera gráfica se puede ver que la curva está, en ciertos puntos, más próxima a los vectores, la mejora no es significativa y el aumento en el número de vectores soporte necesarios sí lo es.

### 2.2.3. Características de las SVM.

En este apartado se resumen algunas de las características que hacen que las SVM sean una buena opción en su aplicación como clasificador o en regresión como se necesita en el desarrollo de esta tesis. Entre estas características se puede destacar [Shawe-Taylor04]:

- La ausencia de mínimos locales es una clara diferencia frente a métodos clásicos como las redes neuronales.
- La dispersión de la solución ("sparseness"), es decir, la reducción que supone el que sólo algunos de los vectores de entrenamiento formen parte de la solución. Esta reducción influye en los requerimientos de memoria y en el número de operaciones cuando se usan en la fase de test, puesto que son inferiores.
- La capacidad de generalización debido a la maximización del margen hace que, aunque el conjunto de entrenamiento tenga pocas muestras, las funciones de decisión o regresión obtenidas sean suficientes para generalizar. Esto es importante en las aplicaciones de esta tesis que usan conjuntos reducidos de entrenamiento sintético.
- El uso de kernel es una característica importante ya que permite elegir entre distintas formas de abordar el problema, con mayor o menor complejidad del kernel según la naturaleza del mismo.
- Los resultados teóricos sugieren que la eficacia de las máquinas de vector soporte se debe a su capacidad de encontrar reglas que clasifiquen objetos con un grado de confianza alto y a su capacidad para evitar el sobre-entrenamiento. Las redes neuronales son capaces de clasificar correctamente casi cualquier conjunto de entrenamiento, pero pueden caer en un sobre-entrenamiento en el que no puedan generalizar correctamente la clasificación.

Obviamente no todo son ventajas sino que existen algunas desventajas en el uso de las SVM que se centran en la lentitud de su aplicación en la fase de test (salvo el caso lineal) [Burges98] y en la dificultad de elegir los parámetros óptimos del kernel o de la regularización.

# Capítulo 3

# Detección de bordes

## 3.1. Estado del arte

Los sistemas de visión automáticos generalmente requieren de un paso inicial, que condiciona el resto de pasos a seguir. Se trata de la segmentación de la imagen en los distintos objetos que la componen. Esta segmentación lo que persigue es la reducción de la información presente en la imagen pero manteniendo, lo mejor posible, la estructura de la misma. De esta forma podremos detectar posteriormente alguna característica que nos interese pero sin necesidad de analizar toda la información de la imagen.

La segmentación se puede realizar de dos maneras básicas: bien agrupando píxeles que tienen alguna característica en común (color, textura) o bien buscando las transiciones en los niveles de gris de la imagen para encontrar las fronteras entre los distintos objetos de la misma. Esto último es lo que conocemos como detección de bordes.

Los bordes son zonas de la imagen con fuertes contrastes de intensidad (aunque a veces estas diferencias no tienen por qué ser muy fuertes). Por tanto, no todos los bordes aparecen en las fronteras de los objetos sino que pueden aparecer en zonas fuertemente texturadas. Sin embargo, en este trabajo lo que nos interesa es detectar los bordes para dividir la imagen en áreas que se correspondan a los diferentes objetos dentro de la misma.

Los bordes se asocian fundamentalmente con las altas frecuencias (cambios bruscos en la señal, en este caso una imagen). Por tanto, podríamos pensar que la forma más directa para detectar los bordes sería el filtrado paso alto en el dominio de Fourier, o bien, su contrapartida espacial mediante la convolución con algún tipo de función equivalente a dicho filtrado. En la práctica, la mayoría de los métodos de detección de bordes se realizan en el dominio espacial porque, generalmente, son menos costosos computacionalmente ([González93]).

La cuestión es ahora cómo crear esos filtros (o máscaras) que vamos a convolucionar con nuestra imagen. La idea en la que se basan dichos filtros parte del hecho de que las derivadas son funciones paso alto y que en un dominio discreto (como es el caso de las imágenes digitales) la derivada se puede aproximar simplemente por una diferencia de muestras adyacentes. Puesto que los bordes se corresponden con fuertes gradientes de intensidad podemos pensar que simplemente calculando derivadas en la imagen es posible "marcar" estos bordes.

Esto último lo podemos ver en el caso unidimensional en la figura 3.1. En dicha figura se muestra la representación unidimensional de cómo se ve un borde en una imagen.

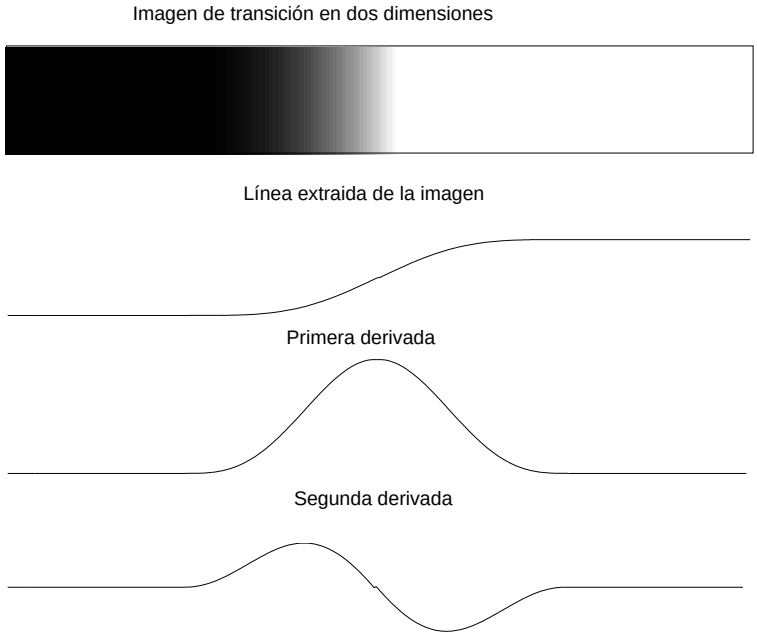



Figura 3.1: Primera y segunda derivadas en un borde representado en una dimensión

Podemos ver que el salto de niveles de gris no es inmediato sino que se produce en una transición más o menos suave. Esto es debido a que las limitaciones de la captura de imágenes (la óptica y la digitalización) hacen que no se puedan registrar esos saltos. En la primera derivada podemos ver como el máximo marca el punto donde se encuentra el borde (justo el punto medio de la transición), en la segunda derivada el paso por cero es el que marca esa posición. Si aplicamos directamente la diferencia entre muestras a una imagen obtendremos algo como lo que se muestra en la figura 3.2. En ella podemos ver cómo al calcular las diferencias en las filas obtenemos marcados los bordes horizontales y en las columnas los verticales. Partiendo de esa idea lo que se busca es una técnica eficiente para calcular la derivada en una imagen bidimensional y, con ella, marcar los bordes.

Como hemos dicho la derivada es aproximada de la siguiente forma:

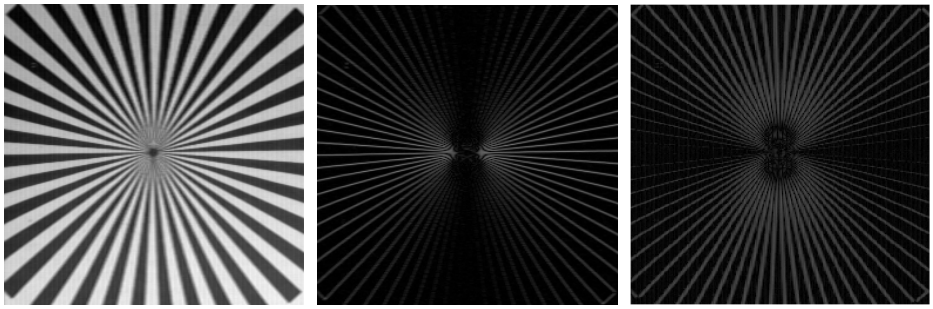
$$\frac{df(i)}{d(i)} \approx f(i+1) - f(i). \tag{3.1}$$

Calcular la derivada de esa manera es equivalente a convolucionar la imagen con $[-1 \quad 1]$, y en el caso de la segunda derivada con $[1 -2 \quad 1]$.

En los siguientes puntos se estudiarán más a fondo técnicas que se basan en la idea anterior aunque cada uno de ellos tiene sus propias peculiaridades.

Podemos decir que hay dos formas comunes de estimar la primera derivada en la detección de bordes, los detectores de bordes de brújula ("compass" en la literatura científica) o de Prewitt y los detectores de bordes de gradiente.

- La detección de bordes de brújula está basada en la convolución de la imagen con un conjunto de 8 máscaras cada uno de los cuales es sensible a una dirección concreta. La



(a) (b) (c)

Figura 3.2: Primera derivada de una imagen ejemplo: (a) Imagen original; (b) Aproximación utilizando diferencias de las filas; (c) Aproximación utilizando diferencias de las columnas

máscara que produce el máximo en un píxel determinado es la que nos proporciona la magnitud y la orientación.

- La detección por gradiente es la técnica más usada de las dos y se basa en la convolución con sólo dos máscaras, una para estimar el gradiente en la dirección $x$, $G_x$, y otra para la dirección $y$, $G_y$. La magnitud del gradiente viene dada por:

$$|G| = \sqrt{G_x^2 + G_y^2}, \tag{3.2}$$

y que normalmente se aproxima por:

$$|G| \approx |G_x| + |G_y| \,. \tag{3.3}$$

Muchas veces sólo se necesita esa magnitud para la detección de bordes, sin embargo, si queremos saber la dirección del mismo podremos obtenerla simplemente con

$$\theta = \arctan\left(\frac{G_y}{G_x}\right). \tag{3.4}$$

Tal y como hemos visto en las figuras 3.1 y 3.2, la obtención de la magnitud del gradiente no aporta información directa sobre la localización de los bordes. Podemos simplemente realizar un umbralizado de la imagen de gradiente o bien buscar los máximos locales para obtener una imagen de borde de un píxel de anchura.

Una técnica que se basa en la segunda derivada de la imagen es el detector de bordes de Marr también conocido como detector de paso por cero que realiza el cálculo mediante la Laplaciana de una gaussiana.

Un problema que tienen todas las técnicas de gradiente presentadas hasta el momento, es la sensibilidad frente al ruido. La razón es que la derivada acentúa las altas frecuencias (filtrado paso alto) y, por tanto, se aumenta el ruido. El detector de bordes de Canny intenta reducir ese problema convolucionando la imagen con un operador de suavizado (gaussiana) antes de calcular la derivada. Algo similar se realiza también en la técnica de Marr.

Otras técnicas que vamos a analizar son las de Rothwell y Bergholm que realizan también algún tipo de filtrado gaussiano pero que tienen variaciones interesantes respecto a la técnica de Canny.

### 3.1.1. Métodos de gradiente

Estos métodos tienen en común el que aproximan el gradiente de una imagen mediante una o varias máscaras. Los más conocidos son los de Roberts, Sobel y Prewitt ([Jain89]).

**Roberts.** El operador de Roberts realiza de forma simple y rápida el cálculo del gradiente espacial en dos dimensiones de una imagen. Como hemos dicho previamente dicho cálculo nos puede ayudar a la hora de detectar los bordes.

Este operador se compone de dos máscaras de convolución que se muestran en la figura 3.3. Puede comprobarse que cada una de ellas es una versión girada 90º de la otra.

| | |
|---|---|
| +1 | 0 |
| 0 | -1 |

**(a)**

| | |
|---|---|
| 0 | +1 |
| -1 | 0 |

**(b)**

Figura 3.3: (a) (b) Máscaras de convolución de Roberts.

Cada máscara se aplica de manera independiente sobre la imagen para obtener el gradiente según cada una de ellas. Después, como ya se ha comentado, se pueden combinar para obtener tanto el módulo como la orientación de dicho gradiente.

La principal ventaja de este operador es que es muy rápido de calcular. Con sólo dos píxeles se puede calcular la salida. Pero el reducido tamaño es también su principal desventaja porque le hace muy sensible al ruido y por ello se pueden perder bordes marcados claramente.

**Prewitt.** El operador de Prewitt también realiza la medida del gradiente en una imagen y, por tanto, se podrá utilizar para la detección de bordes en imágenes. La forma de realizar este cálculo es similar al caso de Roberts pero cambiando las máscaras de convolución a las dos que aparecen en la figura 3.4.

Este operador no suaviza suficientemente el ruido y su implementación requiere más tiempo de procesado que el de Roberts.

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -1 | 0 | +1 |
| -1 | 0 | +1 |

**(a)**

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| +1 | +1 | +1 |

**(b)**

Figura 3.4: Máscaras de convolución de Prewitt. (a) Gradiente en x; (b) Gradiente en y

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

**(a)**

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

**(b)**

Figura 3.5: Máscaras de convolución de Sobel. (a) Gradiente en x; (b) Gradiente en y

**Sobel.** El operador de Sobel realiza una operación similar al de Prewitt pero además añade un efecto de suavizado mayor modificando las máscaras como se muestra en la figura 3.5. El suavizado es interesante porque atenúa parte del ruido, de manera que el efecto amplificador de ruido de la derivación se atenúa también.

Estas máscaras al igual que las de Prewitt están diseñadas para responder de manera máxima a bordes orientados vertical y horizontalmente. Se aplicarán por separado y posteriormente se podrá obtener la magnitud y la orientación.

El operador de Sobel es más lento en su cálculo que el de Roberts pero tiene un efecto de suavizado sobre la imagen y eso hace que sea menos sensible al ruido. La salida de este operador es normalmente mayor, para bordes similares, que la de Roberts.

En imágenes reales el efecto de suavizado de este operador hace que el resultado sea un borde de varios pixeles de grosor. Esto normalmente no es deseable y por tanto hay que realizar un posprocesado que puede ser un simple adelgazamiento o un proceso de histéresis parecido al que se utiliza en el operador de Canny que estudiaremos más adelante.

**Discusión y ejemplos.** En la figura 3.6 se muestra el resultado aplicar cada uno de los operadores presentados sobre una imagen ejemplo. En 3.6(a) y 3.6(b) se muestran la imagen original y otra con ruido gaussiano añadido. En las imágenes 3.6(c), 3.6(d) y 3.6(e) se muestra el módulo calculado según la ecuación (3.2) para cada uno de los operadores aplicados a la imagen sin ruido. En las imágenes 3.6(f), 3.6(g) y 3.6(h) se representa el módulo para la imagen con ruido. Las imágenes de gradiente han sido normalizadas respecto a su valor máximo para poder visualizarlas correctamente.

Podemos apreciar que, en efecto, el operador de Roberts produce gradientes de menos amplitud y que además el ruido le afecta de manera importante por su efecto amplificador sobre el mismo. En el caso de los operadores de Prewitt y Sobel no se aprecia, a simple vista, diferencia alguna aunque teóricamente sí que las hay. La máscara de Prewitt es el resultado de hacer la multiplicación matricial de la ecuación (3.5) mientras que la de Sobel es el resultado de (3.6).

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{3.5}$$

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \tag{3.6}$$

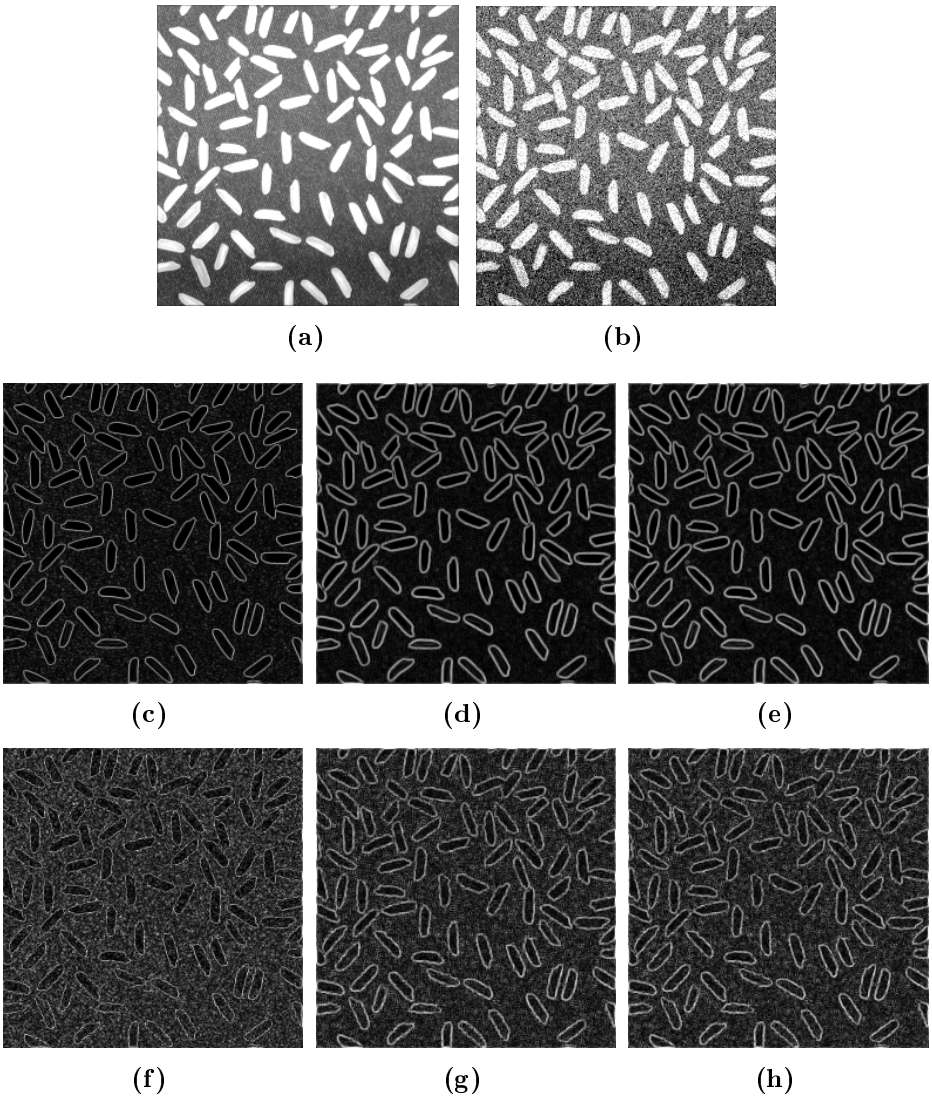



Figura 3.6: Resultado de la aplicación de distintos operadores. (a)Imagen original. (b) Imagen con ruido; (c)(d)(e) Imágenes de bordes aplicando Roberts, Prewitt y Sobel sobre la imagen sin ruido. (f)(g)(h) Imágenes de bordes aplicando Roberts, Prewitt y Sobel sobre la imagen con ruido gaussiano.

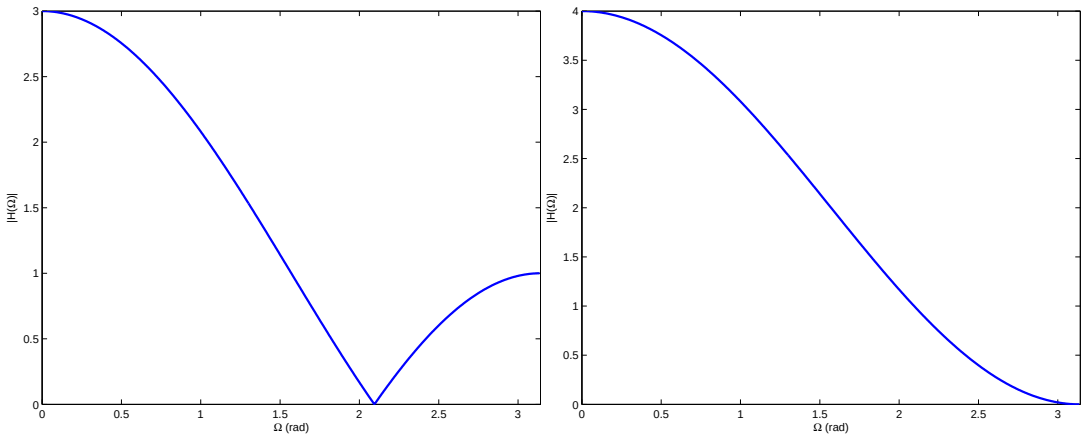



Figura 3.7: (a) Respuesta en frecuencia de [1 1 1] (Prewitt) (b) Respuesta en frecuencia de [1 2 1] (Sobel)

Esto quiere decir que aplicar los operadores de Prewitt y Sobel es equivalente a aplicar primero el operador columna y luego el operador fila según aparecen en las ecuaciones anteriores. El operador columna es un filtro diferenciador y marca los bordes de la imagen. El operador fila supone un suavizado. En la figura 3.7 podemos ver que en el correspondiente al operador de Sobel el suavizado es mayor porque elimina más frecuencias altas que el de Prewitt.

### 3.1.2. Método de brújula

El método de detección de brújula (o compás según la traducción elegida) es una alternativa a los métodos de gradiente estudiados anteriormente. Este método nos dará como resultado dos imágenes, una en la que se obtendrá la magnitud del gradiente y otra que nos dará la orientación de los bordes de la imagen de entrada. Para obtener esas dos imágenes la imagen de entrada se convoluciona con un conjunto de (generalmente 8) máscaras, cada una de las cuales es sensible a bordes en distintas orientaciones. Para cada píxel la magnitud del gradiente local se toma como el máximo de las respuestas de cada una de las máscaras utilizadas. La orientación para ese píxel será la asignada a la máscara que produce el máximo gradiente.

Para este proceso se pueden utilizar varias máscaras. Como ejemplo, en la figura 3.8, mostramos el conjunto de máscaras basadas en la de Prewitt.

El conjunto de las 8 máscaras se consigue partiendo de cualquiera de ellas y rotando circularmente los coeficientes como se puede comprobar en la figura 3.8. Cada una de las máscaras resultantes es sensible a una orientación que puede ir desde $0^o$ hasta $315^o$ en pasos de $45^o$ siendo la orientación de los $0^o$ correspondiente a los bordes verticales.

El detector de bordes de brújula es una buena forma de calcular tanto el gradiente como la orientación de los bordes. El valor del gradiente en cada píxel será el de la máscara que produzca el máximo y la orientación se corresponderá con la orientación asociada a esa máscara que produce ese máximo. Por tanto, para calcular el gradiente necesitamos calcular la convolución con 8 máscaras aunque la orientación es estimada sin ninguna

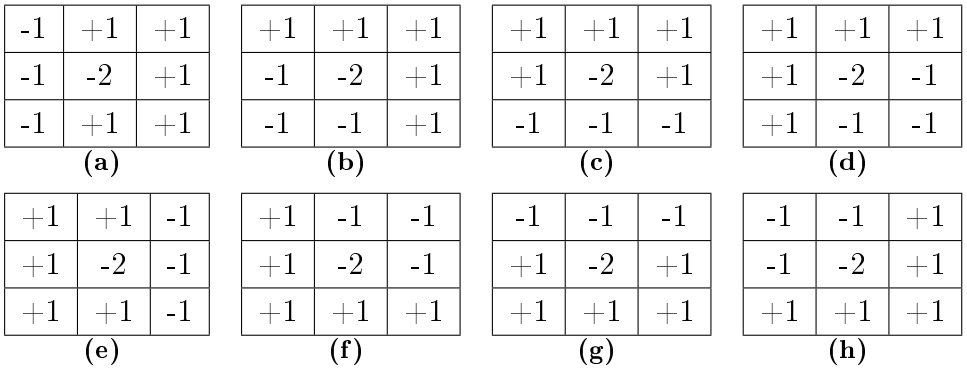



Figura 3.8: Máscaras de Prewitt. (a) 0º (b) 45º (c) 90º (d) 135º (e) 180º (f) 225º (g) 270º (h) 315º

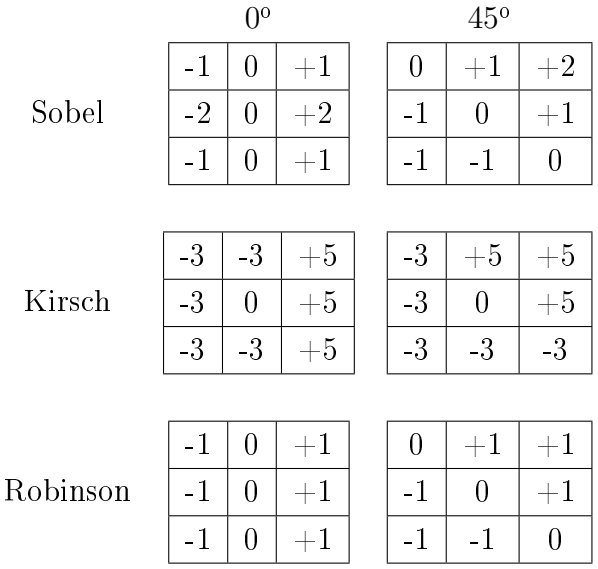
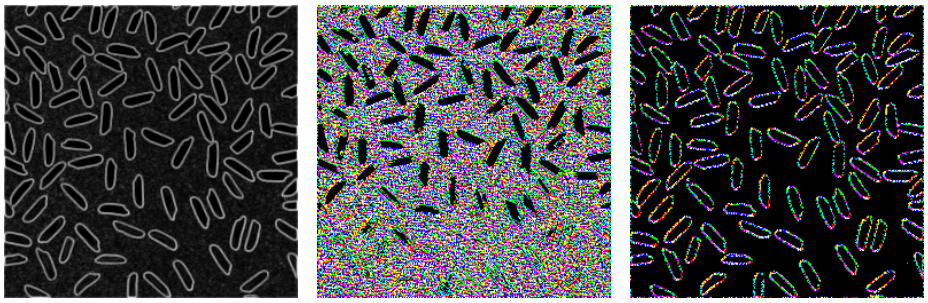



Figura 3.9: Distintas máscaras usadas en la detección de brújula.

operación adicional. En los métodos presentados anteriormente se necesitaba estimar la orientación partiendo de los gradientes vertical y horizontal mediante una operación de arco tangente pero, por contra, sólo necesitábamos la convolución con dos máscaras.

Existen variantes a las máscaras presentadas en la figura 3.8 y que se basan en otros modelos distintos a los de Prewitt. Algunos de ellos los podemos ver en la figura 3.9. Sólo se han representados dos de las máscaras correspondientes a cada tipo pues el resto se obtendrían por rotación de los coeficientes.

El resultado de usar las distintas variantes es similar. La principal diferencia es la escala de magnitud del resultado. La ventaja de usar máscaras como las de Sobel, es que sólo 4 de las 8 máscaras tienen que calcularse pues debido a su simetría las 4 primeras máscaras son justo las opuestas a las 4 siguientes. Por tanto, una vez calculados las 4 primeras convoluciones, las siguientes se obtienen por simple cambio de signo.



(a) (b) (c)

Figura 3.10: Magnitud y orientación usando el operador brújula de Prewitt (a) Magnitud (b) Orientación (c) Orientación cuyo gradiente está por encima de un umbral dado

En la figura 3.10 se puede ver el resultado de la aplicación del método de brújula. En ella podemos ver tanto la magnitud del gradiente obtenida como la orientación en cada pixel. Para facilitar la visualización cada orientación tiene asignado un color. En realidad, de toda la información que aparece en la imagen de orientación sólo es útil la que se corresponda con valores de gradiente suficientemente grandes como para corresponder a un borde. Estos lo podemos ver en la imagen 3.10(c).

### 3.1.3. Método de Laplaciana de una Gaussiana (LoG)

Todos los operadores anteriores, tienen una aproximación hacia derivadas de primer orden sobre el valor de los píxeles en una imagen. Existen métodos que utilizan detectores de bordes basados en derivadas de segundo orden. Uno de los más populares es el Operador Laplaciana:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}.$$

Este operador, para ser aplicado sobre las imágenes, ha de aproximarse mediante una máscara discreta. En la figura 3.11 se muestran algunas aproximaciones de máscaras 3x3.

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| -1 | 2 | -1 |
| 2 | -4 | 2 |
| -1 | 2 | -1 |

Figura 3.11: Diferentes aproximaciones a la laplaciana

Aunque la Laplaciana responde a las transiciones de intensidad, rara vez se utiliza en la práctica para la detección de bordes, porque:

- Los operadores basados en la primera derivada son sensibles al ruido en imágenes. El Laplaciano aún lo es más.

- Genera bordes dobles.
- No existe información direccional de los ejes detectados.

Para minimizar los efectos del ruido, un método consiste en realizar junto con la operación de detección de bordes un proceso de suavizado de la imagen. Uno de los métodos más utilizados es el suavizado por medio de una Gaussiana. Con lo que el proceso sería:

1. Convolucionar la imagen original con un filtro gaussiano.
2. Calcular las derivas sobre la imagen suavizada.

Como ambas operaciones son lineales podemos combinarlas de diferentes formas:

1. Suavizado de la imagen y cálculo de la segunda derivada.
2. Convolución de la imagen original utilizando el laplaciano del Gaussiano (***Operador LoG*** o Sombrero Méjicano).

$$\nabla^2 (G * I) = \nabla^2 G * I,$$

$$LoG(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

Este método de detección de bordes fue propuesto por primera vez por Marr y Hildreth ([Marr80]) quienes introdujeron el principio de detecciones mediante el método de cruces por cero. El principio en que se basa este método consiste en encontrar las posiciones en una imagen donde la segunda derivada toma el valor 0. Los pasos serían:

1. La función gaussiana suaviza o difumina los ejes.
2. Se calcula la segunda derivada de la imagen difuminada.
3. Se detectan los cruces por cero en los bordes.

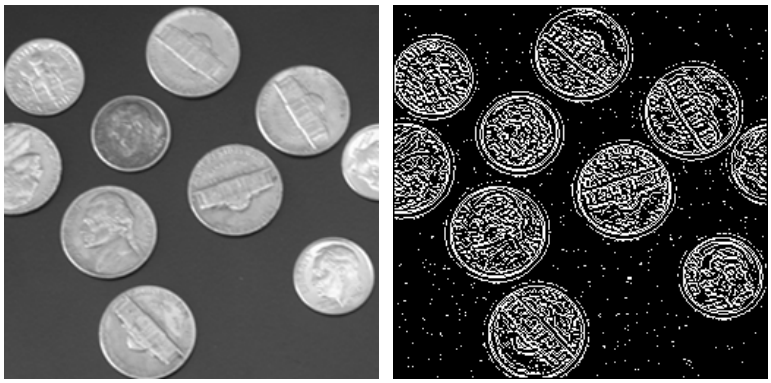
El proceso de difuminación es ventajoso ya que:

- La Laplaciana podría ser infinita en los bordes de la imagen sin suavizar.
- La posición de los bordes se mantiene.

El Operador LoG es también sensible al ruido, pero los efectos del ruido pueden ser reducidos si se ignoran los cruces por cero producidos por pequeños cambios en la intensidad de la imagen. Además, nos da información de la dirección de los ejes, determinada mediante la dirección del cruce por cero.

El operador LoG se puede aproximar, de forma muy efectiva, mediante la convolución con una máscara que es la diferencia de dos máscaras Gaussianas con diferentes valores de desviación estándar. Se le conoce como Diferencia de Gaussianas (DoG).

El operador de Marr-Hildreth llegó a ser uno de los más utilizados por las siguientes razones:



(a) (b)

Figura 3.12: Ejemplo de aplicación de la laplaciana de una gaussiana y su paso por cero.

- Sus fundamentos están basados en los campos receptivos de los ojos de animales.
- El operador es simétrico. Los ejes se encuentran en todas las orientaciones, cosa que no sucede con los operadores basados en la primera derivada, los cuales son direccionales.
- Los cruces por cero de la segunda derivada son más fáciles de determinar que los máximos en la primera derivada. Sólo se necesita detectar un cambio de signo en la señal. Por otro lado, los cruces por cero de una señal se encuentran siempre sobre contornos cerrados.

Sin embargo presenta una serie de problemas:

- La influencia del ruido es considerable en la segunda derivada.
- La generación, siempre, de contornos cerrados no es realista.
- El operador DoG marca puntos considerados como ejes en algunas localizaciones donde no hay bordes.

En la figura 3.12 puede verse un ejemplo de aplicación de esta técnica. En ella pueden verse los efectos negativos de su aplicación, como son el ruido y el doble borde, especialmente evidente en los contornos de las monedas.

### 3.1.4. Método de Canny

Uno de los métodos propuestos para la detección de bordes más conocido, lo propuso J. Canny en [Canny86]. En este caso la detección de bordes se trata como un problema de procesado de señal dirigido a diseñar un operador óptimo en una serie de criterios. Para ello, especificó formalmente una función objetivo que debería de ser minimizada. Dicha función fue utilizada para diseñar el operador. La función objetivo se diseñó de forma que fuera óptima en los siguientes criterios:

- Maximizar la relación señal ruido con objeto de obtener una buena detección. Esta maximización favorece el realce de verdaderos positivos.

- Obtener una buena localización de los bordes.

- Minimizar el número de respuestas sobre bordes simples. Esto favorece la identificación de verdaderos negativos es decir, los puntos correspondientes a no-bordes no son marcados

Después de un análisis, Canny determinó que la función objetivo se podía describir como la suma de cuatro términos exponenciales. Al final, esta función presenta un gran parecido con la primera derivada de una Gaussiana, así que ésta es la que se utiliza como aproximación mejor a la función óptima.

El procedimiento general para la detección de bordes es como sigue:

1. Obtener los máximos de las derivadas parciales de la función imagen ($I$) en las direcciones ortogonales a las direcciones de los bordes y suavizar la señal a lo largo de las direcciones de los mismos. Por tanto, el operador de Canny busca los máximos en la siguiente función:

$$\frac{\partial^2}{\partial n^2}\left(G * I\right); n = \frac{\nabla G * I}{|\nabla G * I|}.$$

   Hay varios métodos para la implementación real de este proceso. Uno de ellos consiste en convolucionar la imagen con una gaussiana y buscar máximos en las derivadas parciales de la imagen transformada (utilizando máscara parecidas a las de Sobel). El valor más alto que se produce en una cierta dirección sobre un píxel se almacena. Así, se guardan los resultados de la convolución y la dirección del borde para cada punto.

2. Cualquier valor de gradiente que no es un pico local se pone a 0. La información con respecto a la dirección de los bordes se utiliza en este proceso.

3. Encontrar conjuntos de puntos de borde conectados.

4. Umbralizar dichos bordes para eliminar los bordes insignificantes. Canny introduce la idea de "Proceso de Histéresis". Se introducen dos umbrales. Considerando un segmento de línea, si un punto presenta un valor de gradiente superior al umbral superior, se acepta inmediatamente como punto de borde. Si ese valor es más pequeño que el umbral inferior, el punto en cuestión es desestimado. Puntos cuyo valor de gradiente se encuentra entre los dos umbrales son considerados como bordes si se encuentran conectados a puntos que ya han sido aceptados como tales. Esto viene a significar que, cuando empezamos un borde, no paramos hasta que el gradiente tiene un valor muy pequeño.

En la figura 3.13 se puede ver un ejemplo de la aplicación del algoritmo de Canny sobre una imagen real. Es evidente que los resultados obtenidos con este algoritmo son fuertemente dependientes de los valores de $\sigma$ y los umbrales establecidos. Por tanto, no hay una única salida para este algoritmo sino que en cada aplicación se emplearán unos parámetros determinados para obtener los mejores resultados.

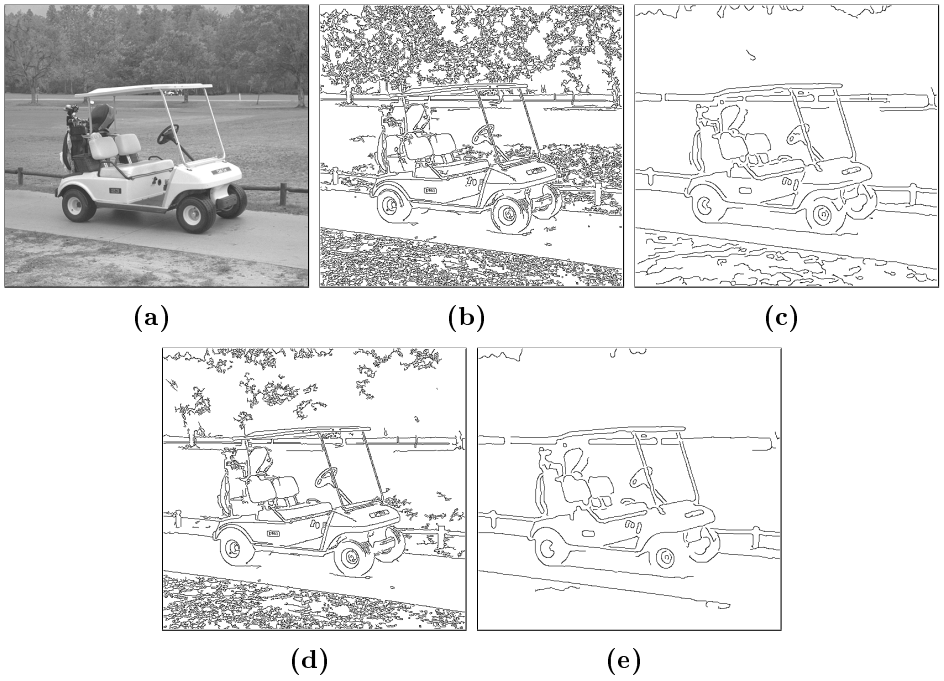



Figura 3.13: Ejemplo de detección de bordes usando el algoritmo de Canny. (a) Imagen original. (b) Detección con $\sigma = 0.6$ y umbrales 0.3 y 0.8. (c) Detección con $\sigma = 2.4$ y umbrales 0.3 y 0.8. (d) Detección con $\sigma = 0.6$ y umbrales 0,3 y 0,9. (c) Detección con $\sigma = 2.4$ y umbrales 0.3 y 0.9.

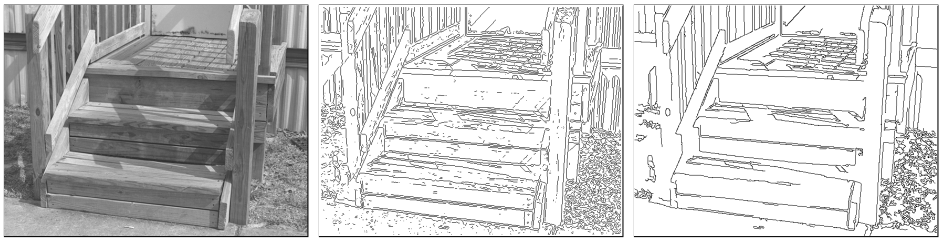
### 3.1.5. Método de Rothwell

El algoritmo de Rothwell [Rothwell95] emplea umbralizado dinámico, por tanto, varía el umbral de detección de borde a lo largo de la imagen. El algoritmo es muy similar al de Canny ya que utiliza una función gaussiana para suavizar, seguida por una diferenciación. La principal discrepancia entre los dos algoritmos es que el algoritmo de Rothwell realiza un adelgazamiento de los bordes como un proceso posterior a la detección y además usa un umbralizado dinámico en vez de histéresis. La razón de no usar la supresión de máximos y sí un adelgazamiento es que la supresión falla en las uniones de los bordes debido al proceso de suavizado. La razón para no usar la histéresis es la creencia de que la fuerza de un borde no tiene ninguna importancia particular para el proceso posterior (por ejemplo reconocimiento de objetos).

Algunos trabajos sobre medidas de calidad de detección de bordes como [Heath97] o [Dougherty98] dan resultados similares para los algoritmos de Canny y Rothwell. En la figura 3.14 se presenta un ejemplo en el que se pueden comparar visualmente ambos algoritmos.

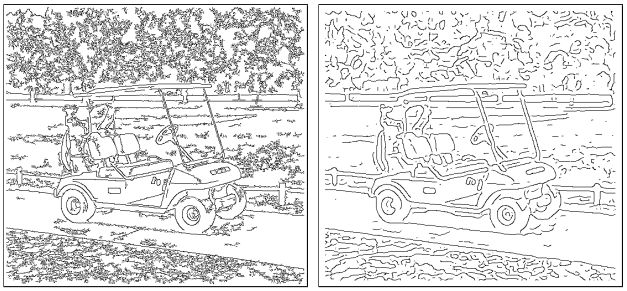
### 3.1.6. Método de Bergholm

El método de Bergholm ([Bergholm87]) presenta una novedad en relación con los estudiados hasta ahora puesto que la detección de bordes se realiza en un espacio multiresolución. Los bordes se detectan primero en un espacio cuya resolución es menor utilizando un algoritmo similar al de Canny, esto es, suavizando la imagen con un filtro gaussiano para



(a) (b) (c)

Figura 3.14: Comparación de los algoritmos de Canny y Rothwell. (a) Imagen original. (b) Rothwell con $\sigma = 1$ y umbrales 13 y 0.9. (c) Canny con $\sigma = 1$ y umbrales 0.3 y 0.9.



(a) (b)

Figura 3.15: Ejemplo de funcionamiento del algoritmo de Bergholm (a) Bordes detectados con un barrido desde $\sigma = 4$ hasta $\sigma = 0.5$ y un umbral de 10. (b) Bordes detectados sin barrido con una $\sigma = 4$ y un umbral de 10.

después realizar el gradiente. Por último, se buscan en el gradiente los máximos locales que a la vez están por encima de un determinado umbral. Posteriormente se refina la búsqueda en las zonas donde hay bordes (enfocado de bordes) mediante la búsqueda de los mismos en un espacio de mayor resolución. Esto se consigue suavizando la imagen con una gaussiana más pequeña (con menor anchura). De esta forma los bordes detectados a una resolución menor sirven de guía para la búsqueda posterior a mayores resoluciones.

En [Heath97] este algoritmo obtiene muy buenos resultados que son, en algunos casos, mejores que los de Canny. En la figura 3.15 se puede ver un ejemplo de aplicación del mismo. En 3.15(a) se aprecia un buen número de detalles puesto que se aplica el enfocado de los bordes mientras que en 3.15(b) no se realiza dicho enfocado y se observa que los detalles se pierden.

## 3.2. Detección de Bordes usando las Máquinas de Vectores Soporte

En esta sección se propone, como alternativa a las técnicas de detección de bordes existentes, el uso de las SVM para clasificar los píxeles de una imagen entre aquellos pertenecientes a un borde y los que no. Realmente, ese es el proceso que realizan los detectores de borde aunque no sean explícitamente clasificadores.

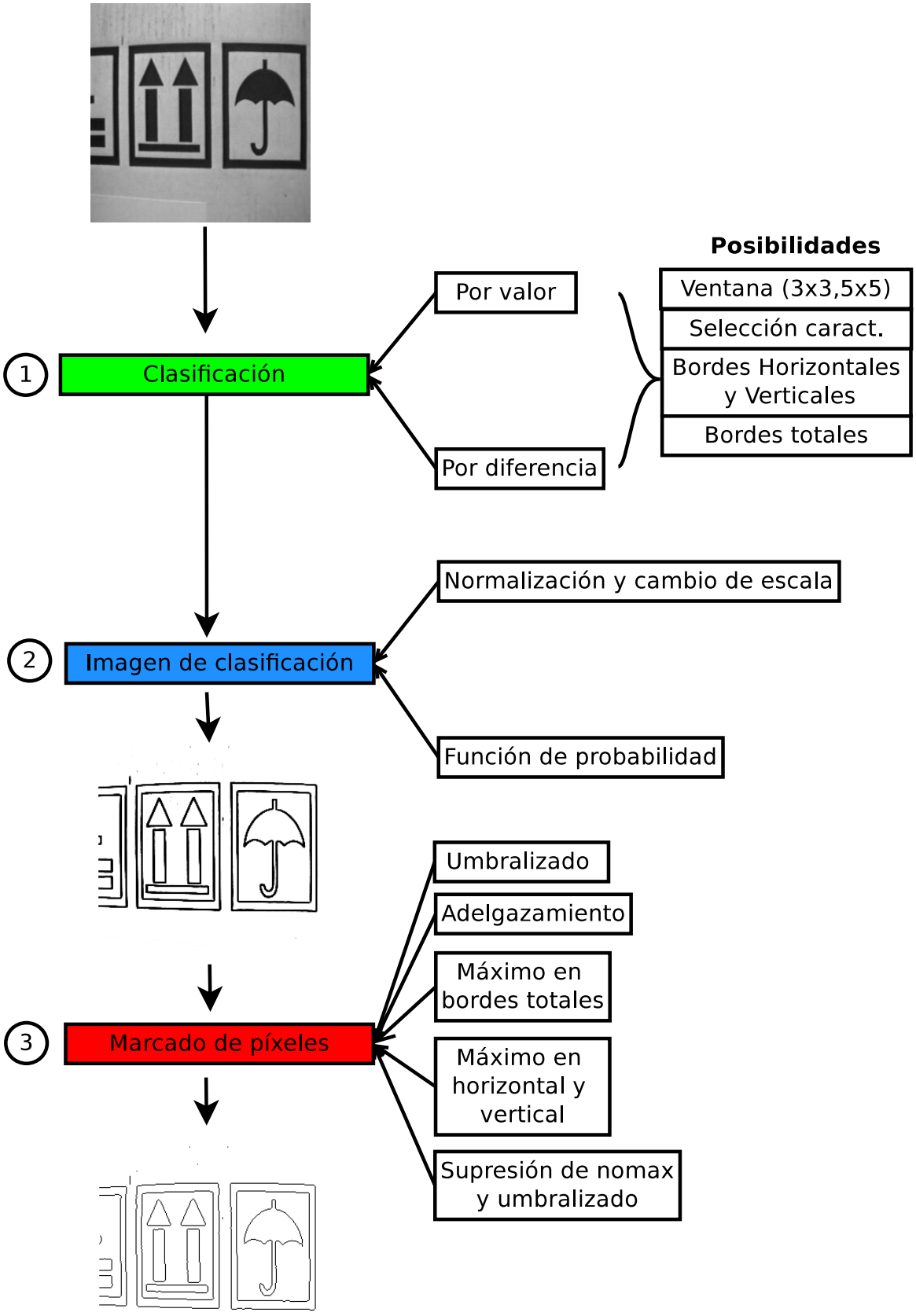



Figura 3.16: Esquema de las posibilidades para la detección de bordes

| | | |
|---|---|---|
| $x-1, y-1$ | $x-1, y$ | $x-1, y+1$ |
| $x, y-1$ | $\boldsymbol{x, y}$ | $x, y+1$ |
| $x+1, y-1$ | $x+1, y$ | $x+1, y+1$ |

Figura 3.17: Entorno $3 \times 3$ de un píxel $x, y$.

En el estudio del estado del arte se ha visto que los bordes son zonas de la imagen donde se producen transiciones en niveles de gris, por tanto, para clasificar los bordes no sólo es necesario el píxel a clasificar sino que es necesaria también información de los píxeles adyacentes. Esta información será la utilizada para la obtención de un vector que deberá ser clasificado por las SVM como borde o no borde. Al realizar dicha operación se observará que no se obtiene directamente una imagen con los bordes marcados en un anchura de un píxel sino que la naturaleza difusa de los bordes hará que sea necesario un postprocesado similar a los vistos en el estado del arte (Ver por ejemplo el método de Canny en pag. 33). Esto deja un esquema de aplicación como el mostrado en la figura 3.16. En el aparecen tres fases diferenciadas, una primera en la que se clasifican los píxeles de la imagen, una segunda en la que, a partir de la información anterior se obtiene la imagen de clasificación y una tercera en la que se produce el marcado definitivo de los píxeles.

En todas las fases se proponen distintas posibilidades para dar mayor flexibilidad al método propuesto. En la fase de clasificación de píxeles se podrán elegir, sobre todo la forma en que se genera el vector a clasificar pero también el tipo de borde a detectar. Con ello se pueden dar las siguientes alternativas:

- Tipo de vector. El vector a clasificar puede contener distinta información de la imagen a procesar. En esta tesis se propone utilizar o bien los valores de gris de la imagen sin ningún tipo de modificación o bien los valores de las diferencias del valor de gris del píxel a clasificar y sus adyacentes. Esta última versión permite obtener vectores con una componente menos y con información relevante cuando se trata de detectar bordes, en los que la diferencia de gris entre zonas es lo más importante.

- Tamaño de la ventana alrededor del píxel a clasificar. Se puede definir una ventana alrededor del píxel indicando los valores de la imagen que formarán parte del vector a clasificar. Generalmente los métodos de detección usan una ventana de $3 \times 3$ (Ver figura 3.17) pero puede ser de $5 \times 5$ o mayor. A priori, mayor tamaño implica más características en el vector y mayor coste computacional aunque puede suponer una mejora en la detección al aumentar la información para la clasificación.

- Una posibilidad es intentar reducir el número de características de los vectores a clasificar mediante las técnicas de selección de características propuestas para ello. De esa manera se pueden mantener los resultados en gran manera pero reducir significativamente el coste computacional.

- Por otro lado se pueden detectar bordes en todas las direcciones posibles (Bordes totales) o sólo los horizontales o verticales como en algunos de los métodos estudiados en el estado del arte. En el estudio aquí presentado se han utilizado ambas, ya que el postprocesado posterior es distinto y se propone explorar las distintas posibilidades.

En esta primera fase es donde se han introducido las mayores innovaciones puesto que el uso de las SVM es novedoso en la clasificación de los bordes. La fase siguiente está muy relacionado con esta, porque se utiliza la información de clasificación para obtener una imagen que podemos comparar con las obtenidas en el estado del arte al aproximar el gradiente. Esta imagen será la utilizada en la última fase de marcado. Las posibilidades en esta caso son dos:

- Utilizar la salida de la función de clasificación de la SVM (no su signo) para, mediante normalización y cambio de escala, obtener una imagen en escala de grises.
- Convertir la salida de la función de clasificación en una función de densidad de probabilidad y, a partir de ella, obtener una imagen en escala de grises. En este caso la escala de grises dará directamente una idea de la probabilidad de que un píxel sea borde o no.

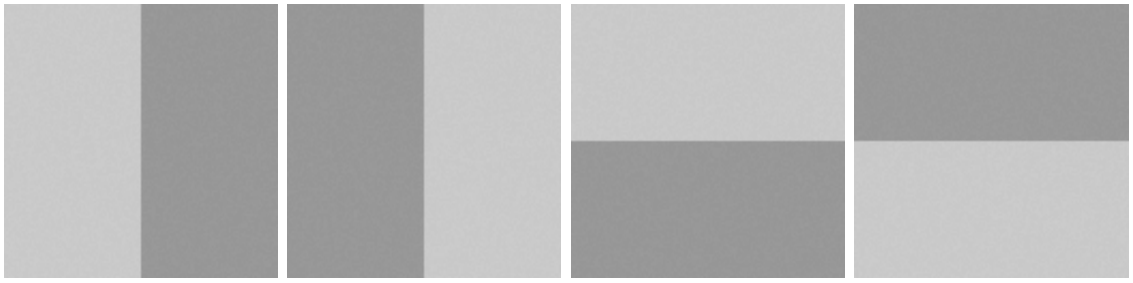
Ambos casos se tratan ampliamente en la sección 3.2.3.

La tercera fase, de marcado de los píxeles, utiliza algunas técnicas ya introducidas en el estado del arte y algunas modificaciones novedosas necesarias para su optimización. Las posibilidades en el caso del marcado de los píxeles serán las siguientes:

- Umbralizado. Con las técnicas de obtención de gradientes del estado del arte, el marcado de los píxeles con un simple umbralizado no permite obtener unos bordes definidos. Sin embargo, mediante el uso de SVM y un simple umbralizado es posible la obtención de un marcado de bordes, sobre todo en el caso de usar las diferencia de valores.
- Se puede utilizar directamente un umbralizado, para después refinarlo mediante técnicas de adelgazamiento (thinning).
- Otra opción es buscar los máximos obtenidos en la clasificación y marcarlos como bordes de la imagen. Esta técnica es sencilla pero necesita un paso posterior de limpieza de píxeles que también se realizará con SVM.
- Si los máximos se buscan por separado en los bordes horizontales y verticales los resultados son también aceptables y no se necesita el paso posterior de limpieza. Sin embargo se necesitan dos modelos y dos pasos de clasificación.
- Si se dispone de los bordes horizontales y verticales se pueden utilizar las técnicas de supresión de no máximos y de umbralizado con histéresis del método de Canny.

### 3.2.1. Entrenamiento.

Puesto que queremos usar las SVM para clasificar los píxeles, un paso importante es el del entrenamiento. En el entrenamiento se ha de indicar qué vectores son considerados como bordes y cuales no, para que, a partir de ahí se infiera la función de clasificación mediante el entrenamiento. Además, se ha de elegir convenientemente el kernel a utilizar en las SVM porque dicho kernel influirá en la calidad de la detección y en el número de vectores soporte a utilizar. Una vez elegido el kernel habrá que elegir los parámetros para



**(a)** ***Entrenamiento de bordes verticales*** **(b)** ***Entrenamiento de bordes horizontales***

Figura 3.18: Ejemplo de imágenes sintéticas entrenamiento. Tienen un nivel de gris para la parte más clara de 200, para la más oscura de 150, un tamaño de $128 \times 128$ y un nivel de ruido de 5.

el entrenamiento como son el factor regularización $C$ y, dependiendo del kernel, $\gamma$, grado del polinomio, etc.

Hay que indicar que en esta sección se ha utilizado el vector de valores alrededor del píxel a clasificar y no el de diferencias. En el caso de utilizar diferencias los procedimientos serían similares aunque cambiaría el tamaño del vector (en una unidad) y los valores utilizados serían de diferencia de valor de gris entre el píxel central y los pertenecientes a las ventanas. En el último punto de esta sección se trata este tipo de entrenamiento.

Es importante recalcar que las imágenes de bordes que aparecen en esta sección sólo muestran la clasificación de los píxeles directamente. Es decir, no hay ningún tipo de procesado posterior a la clasificación. Según el esquema de la figura 3.16, se realiza sólo el paso 1. Esto es así para poder observar directamente los efectos del entrenamiento sin ningún proceso posterior que pueda enmascararlos.

**Imágenes sintéticas de entrenamiento.** Una posibilidad para el entrenamiento es generar lo que se conoce como "ground thruth", es decir, marcar en una imagen real los bordes de manera subjetiva y entrenar con esa información, obviamente esto impide el entrenamiento de manera automática y añade al entrenamiento la subjetividad del sujeto que marca los bordes. Otra posibilidad sería utilizar para marcar los bordes alguno de los detectores de bordes presentados en el apartado del estado del arte, de manera que la obtención del entrenamiento sea automática. Este método se descartó porque el entrenar con un método ya establecido no parece eficiente puesto que el trabajo de detección ya estaría realizado por dicho método sin aportar mejoras ni nuevas características. Finalmente se optó por la generación de imágenes sintéticas de entrenamiento, de manera que la localización de los bordes sea conocida a priori. Este método tiene la ventaja de poder generar el entrenamiento de acuerdo a las necesidades del usuario en cada momento, por ejemplo, se puede añadir ruido de un nivel determinado, se pueden usar unos niveles de gris determinados, etc. Esto permite un mayor control sobre el entrenamiento realizado. Una versión inicial de este procedimiento aparece en [Gómez-Moreno01b].

En la figura 3.18 se muestra un ejemplo de imágenes de entrenamiento con las que se pretende la detección de bordes verticales y horizontales. En la generación de dichas imágenes hay que fijar los niveles de gris para la parte más clara y la más oscura, así como su tamaño. Obviamente, si se quiere realizar un entrenamiento válido para la detección

del mayor número de bordes posible, se deben generar varios grupos de imágenes como las del ejemplo en las que los niveles de gris de las zonas clara y oscura sean diferentes, tratando de barrer al máximo la escala de grises. También, es necesario añadir algo de ruido (Gaussiano según la literatura sobre detección de bordes) a la imagen para que el entrenamiento con estas imágenes sea más cercano a la realidad, para incluir mayor número de vectores de entrenamiento distintos y para conseguir cierta inmunidad frente al ruido. El nivel de ruido añadido es un parámetro más en el entrenamiento.

Por tanto, los parámetros a fijar en la generación de imágenes de entrenamiento serán:

1. El tamaño de las imágenes. Más tamaño supone más vectores diferentes para entrenar.

2. La diferencia entre los niveles de gris de las zonas clara y oscura.

3. El nivel de ruido gaussiano.

4. El número de imágenes a generar. Cada imagen debería tener una diferencia entre zonas distinta. A más imágenes más cercanía a la realidad.

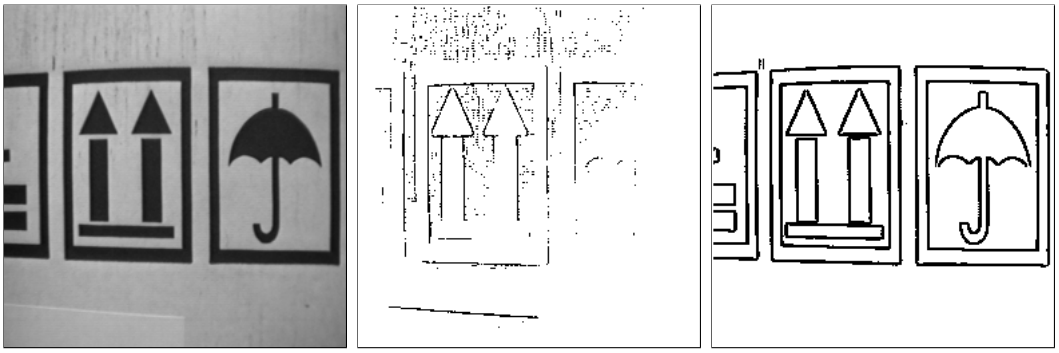
Para las pruebas de idoneidad de las imágenes de entrenamiento se ha utilizado un kernel gaussiano (Ecuación 3.7) en el que sólo se necesita ajustar el valor de $\gamma$ y que produce buenos resultados en gran número de aplicaciones. El parámetro $\gamma$ realiza dos funciones, por un lado permite la normalización de los valores de gris tomados directamente de la imagen y por otro permite distintos grados de generalización en el aprendizaje. Un valor pequeño indica mayor generalización (mayor anchura de la gaussiana) y un valor mayor implica menos generalización o más adecuación del entrenamiento a los ejemplos utilizados para el mismo.

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2). \tag{3.7}$$

La figura 3.19(b) muestra el resultado de la clasificación de bordes utilizando únicamente para entrenar las imágenes de entrenamiento de la figura 3.18 con una ventana alrededor de cada píxel de $3 \times 3$. En la ventana se han tomado los valores de gris de los píxeles para formar los vectores de entrenamiento. La anchura de la gaussiana se ha fijado a un valor $\gamma = 8 \cdot 10^{-5}$. Puede comprobarse que el entrenamiento es pobre ya que no se consigue la clasificación de todos los bordes de la imagen.

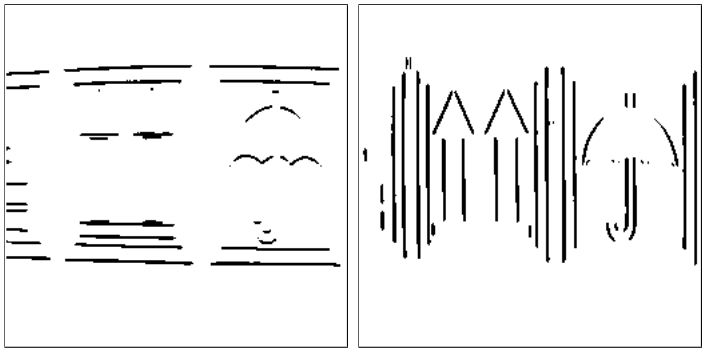
Sin embargo, la imagen 3.19(c) muestra la clasificación de bordes obtenida mediante un entrenamiento similar al anterior pero añadiendo más niveles de gris. Concretamente el entrenamiento se ha realizado usando 40 grupos de imágenes con valores aleatorios de los niveles de gris de las zonas clara y oscura pero manteniendo una diferencia entre las dos zonas de 50 y un nivel de ruido de 5. El tamaño de las imágenes sigue siendo de $128 \times 128$ y la ventana de valores alrededor del píxel de $3 \times 3$, la anchura de la gaussiana es de $\gamma = 8 \cdot 10^{-5}$. Puede comprobarse como en este caso sí que se marcan correctamente los bordes de la imagen y el resultado ha mejorado considerablemente. Es de resaltar el hecho de que un entrenamiento tan sencillo produce una buena generalización en la clasificación de los bordes puesto que aunque sólo se ha entrenado con bordes verticales y horizontales, aparecen bordes en otras direcciones.

En el estudio del estado del arte (Sección 3.1.1) se mostró que los métodos de gradiente realizan la detección de bordes separando por un lado los horizontales y por otro los



(a) (b) (c)

Figura 3.19: Ejemplo de detección de bordes. (a) Imagen original, (b) Bordes obtenidos con un sólo grupo de imágenes de entrenamiento (Figura 3.18), (c) Bordes obtenidos con 40 grupos de imágenes.



(a) (b)

Figura 3.20: Ejemplo de detección de bordes horizontales y verticales. (a) Bordes horizontales, (b) Bordes verticales.

verticales. El método presentado en esta sección también puede realizar esa operación sin más que adecuar el entrenamiento. En la figura 3.20 se presenta un ejemplo de este proceso. La imagen de la figura 3.20(a) se ha obtenido entrenando para la detección exclusiva de bordes horizontales. Para ello se han usado imágenes de entrenamiento como las presentadas anteriormente pero en las que sólo se etiquetan como bordes los de las imágenes horizontales. La imagen de la figura 3.20(b) presenta un proceso similar pero con entrenamiento vertical. Estos ejemplos demuestran la flexibilidad que se puede obtener usando la detección de bordes con SVM sin más que modificar el entrenamiento sintético utilizado.

**Elección de kernel.** Un punto importante en el diseño de aplicaciones con SVM es la elección del kernel. El kernel y el ajuste de los parámetros a él asociados harán que el clasificador realizado sea más o menos efectivo. Existen en la literatura al respecto varios kernel que cumplen las condiciones para ser usados con las SVM [Souza10, Gunn98]. Algunos están especialmente diseñados para la regresión como los ANOVA o los basados en splines y otros se utilizan más en clasificación. Una selección de estos últimos, que han sido probados para la detección de bordes, aparecen en la tabla 3.1

Tabla 3.1: Distintos kernel probados en la detección de bordes.

| **Nombre** | **Expresión** |
|---|---|
| Kernel Polinomial | $k(\mathbf{x_i}, \mathbf{x_j}) = (\gamma \mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$ |
| Kernel Gaussiano | $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \Vert \mathbf{x_i} - \mathbf{x_j} \Vert^2) = \exp\left(-\frac{\Vert \mathbf{x_i} - \mathbf{x_j} \Vert^2}{2\sigma^2}\right)$ |
| Kernel ERBF | $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \Vert \mathbf{x_i} - \mathbf{x_j} \Vert)$ |
| Kernel Sigmoidal | $k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\gamma \mathbf{x_i} \cdot \mathbf{x_j} + c)$ |

Para la aplicación que se busca en esta tesis, lo que importa es obtener los mejores resultados en cuanto a la detección de bordes, con el menor número posible de vectores soporte (mayor velocidad). En las pruebas preliminares se descartó directamente el kernel sigmoidal porque no ofrecía buenos resultados de detección de bordes. Los tres restantes sí producen buenos resultados pero el kernel ERBF genera un número excesivamente alto de vectores soporte sin aportar mejoras, por lo que también se descartó. Finalmente se hicieron pruebas de comportamiento con los Kernel Polinomial y Gaussiano (considerado como estándar en muchas aplicaciones [Hsu03]) para distintos valores de sus parámetros. Para las pruebas se utilizaron imágenes de entrenamiento sintéticas con una diferencia de 50 y un nivel de ruido de 5.

En la figura 3.21 se muestra un ejemplo de dichas pruebas con la imagen Lenna para distintos valores de los parámetros $\gamma$ y $d$ en el kernel polinomial. En dicha imagen se puede comprobar que el efecto de $\gamma$ en este kernel es reducido y se limita a realizar una normalización de valores. Cuando es más pequeño se reduce el número de detalles marcados como se puede comprobar para $\gamma = 5 \cdot 10^{-5}$ en comparación con valores mayores. Es para valores crecientes de $d$ ($d$ controla el grado de generalización del kernel) cuando se aprecia más efecto sobre los bordes en la imagen, cuanto mayor es el valor más detalles aparecen marcados. El número de vectores soporte en estas pruebas se mantuvo relativamente bajo, entre 6 y 16.

En la figura 3.22 se muestran resultados para el kernel gaussiano. En este caso hay un único parámetro a ajustar, $\gamma$, siendo este el encargado de realizar tanto la normalización de valores como la generalización del kernel. Los resultados muestran que tanto para valores bajos como altos de $\gamma$ los bordes marcados son pobres en comparación con lo del kernel polinomial, con la mejor opción en valores cercanos a $10^{-4}$ y $5 \cdot 10^{-5}$. El número de vectores soporte en este caso fluctúa entre 639 y 15 siendo de 75 para los mejores resultados.

Comparando resultados de ambos kernel se puede apreciar que si lo que se necesita es más velocidad se debería elegir el kernel polinomial aunque en su contra se puede decir que es más sensible al ruido, como se puede comprobar viendo el número de puntos aislados en los dos casos. Por otro lado, tal y como se planteó en el apartado de entrenamiento, la diferencia de valor de gris entre las zonas claras y oscuras deberían marcar la sensibilidad del detector de bordes, sin embargo, en el caso del kernel polinomial su sensibilidad es mayor, quizás excesiva, mientras que el gaussiano se acerca más al resultado esperado. Con los resultados presentados podemos llegar a la conclusión de que tanto el kernel

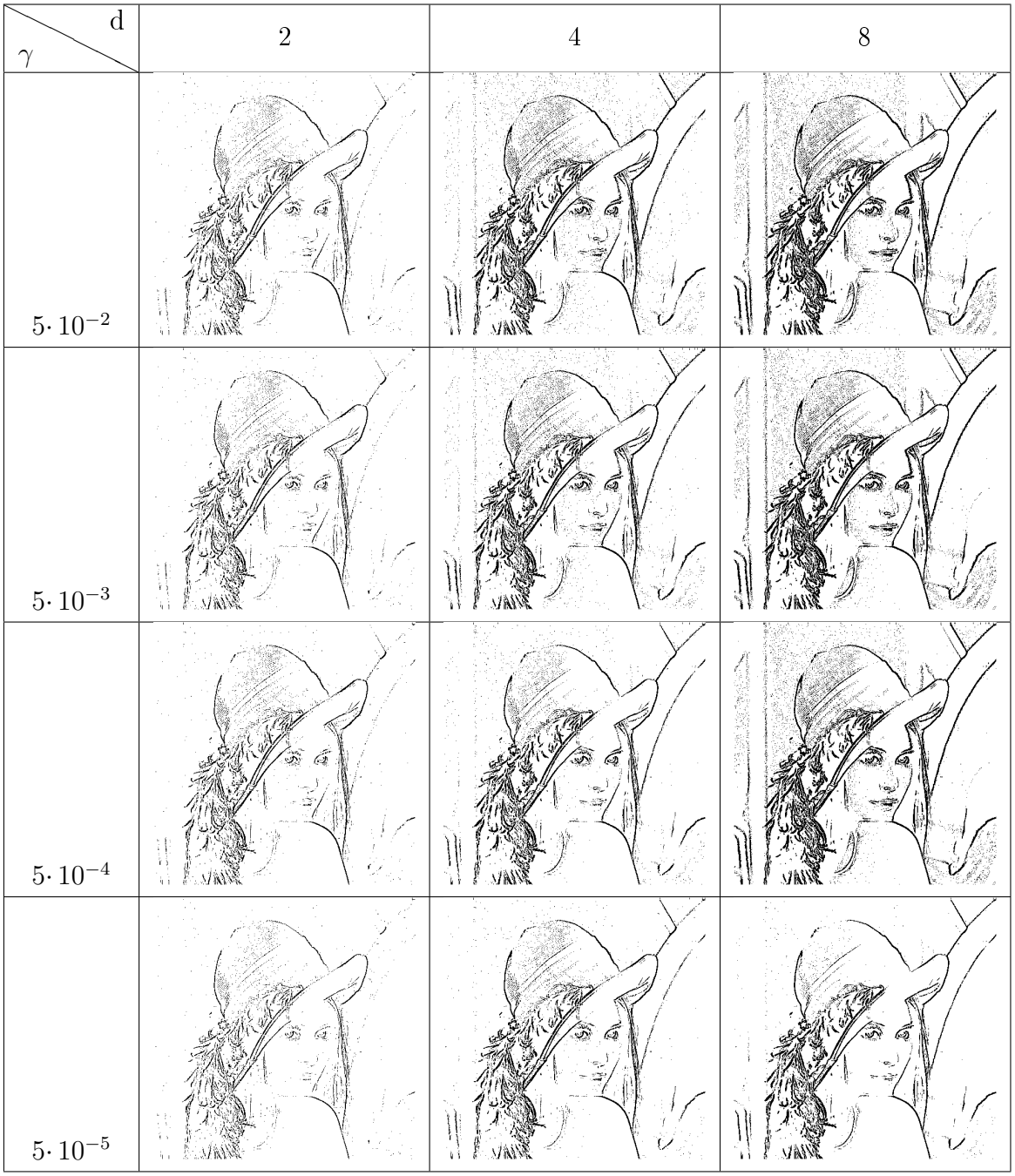



Figura 3.21: Pruebas realizadas usando el kernel polinomial con distintos valores de $\gamma$ y d.

polinomial como el gaussiano son válidos para la detección de bordes aunque con distintas características.

**Selección de características.** Una vez presentado el método de entrenamiento elegido y hechas las primeras pruebas para comprobar la efectividad del método de detección de bordes, es necesario definir el vector de entrada al clasificador de bordes. En las pruebas de la sección anterior los vectores de entrada estaban formados por todos los valores de los píxeles en una ventana de $3 \times 3$ alrededor del píxel a clasificar. Sin embargo, puede que no todos ellos sean necesarios para la clasificación de bordes, y si lo que se pretende es realizar sólo la clasificación de los bordes verticales y horizontales por separado habría que definir cuales de los píxeles son relevantes y cuales no. Un trabajo previo sobre sobre este tema puede encontrarse en [Gómez-Moreno02]

Tabla 3.2: Puntuación obtenida por cada una de las componentes del vector $3\times3$ utilizando la técnica de selección de características presentada en [Chen06].

| | **Completo** | **Vertical** | **Horizontal** |
|---|---|---|---|
| 1 | 0.000002 | 0.000000 | 0.000002 |
| 2 | **0.008162** | 0.000008 | **0.017575** |
| 3 | 0.000010 | 0.000017 | 0.000002 |
| 4 | **0.007756** | **0.013529** | 0.000001 |
| 5 | **0.031405** | **0.013312** | **0.017668** |
| 6 | **0.008033** | **0.013274** | 0.000000 |
| 7 | 0.000001 | 0.000000 | 0.000006 |
| 8 | **0.007466** | 0.000000 | **0.017958** |
| 9 | 0.000000 | 0.000001 | 0.000000 |

Para realizar esta selección de características se ha utilizado la herramienta "fselect" presentada en [Chen06] y que implementa la medida conocida como "F-score". Esta medida se ha elegido ya que es sencilla, se obtienen buenos resultados y está especialmente indicada cuando se usan SVM. Según lo presentado en [Chen06], "F-score" mide la discriminación entre dos conjuntos de números reales. Dados unos vectores de entrenamiento $\mathbf{x}_k, k = 1, \cdots, m$, y si llamamos $n_+$ y $n_-$ al número de ellos que pertenecen a la clase positiva y negativa respectivamente, entonces el "F-score" de la $i_{\text{ésima}}$ característica se define como:

$$F(i) = \frac{\left(\bar{\mathbf{x}}_i^{(+)} - \bar{\mathbf{x}}_i\right)^2 + \left(\bar{\mathbf{x}}_i^{(-)} - \bar{\mathbf{x}}_i\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(x_{k,i}^{(+)} - \bar{\mathbf{x}}_i^{(+)}\right) + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(x_{k,i}^{(-)} - \bar{\mathbf{x}}_i^{(-)}\right)}, \tag{3.8}$$

donde $\bar{\mathbf{x}}_i$, $\bar{\mathbf{x}}_i^{(+)}$, $\bar{\mathbf{x}}_i^{(-)}$ son las medias de la $i_{\text{ésima}}$ característica de los conjuntos total, positivo y negativo respectivamente; $x_{k,i}^{(+)}$ es la $i_{\text{ésima}}$ característica del $k_{\text{ésimo}}$ vector positivo y $x_{k,i}^{(-)}$

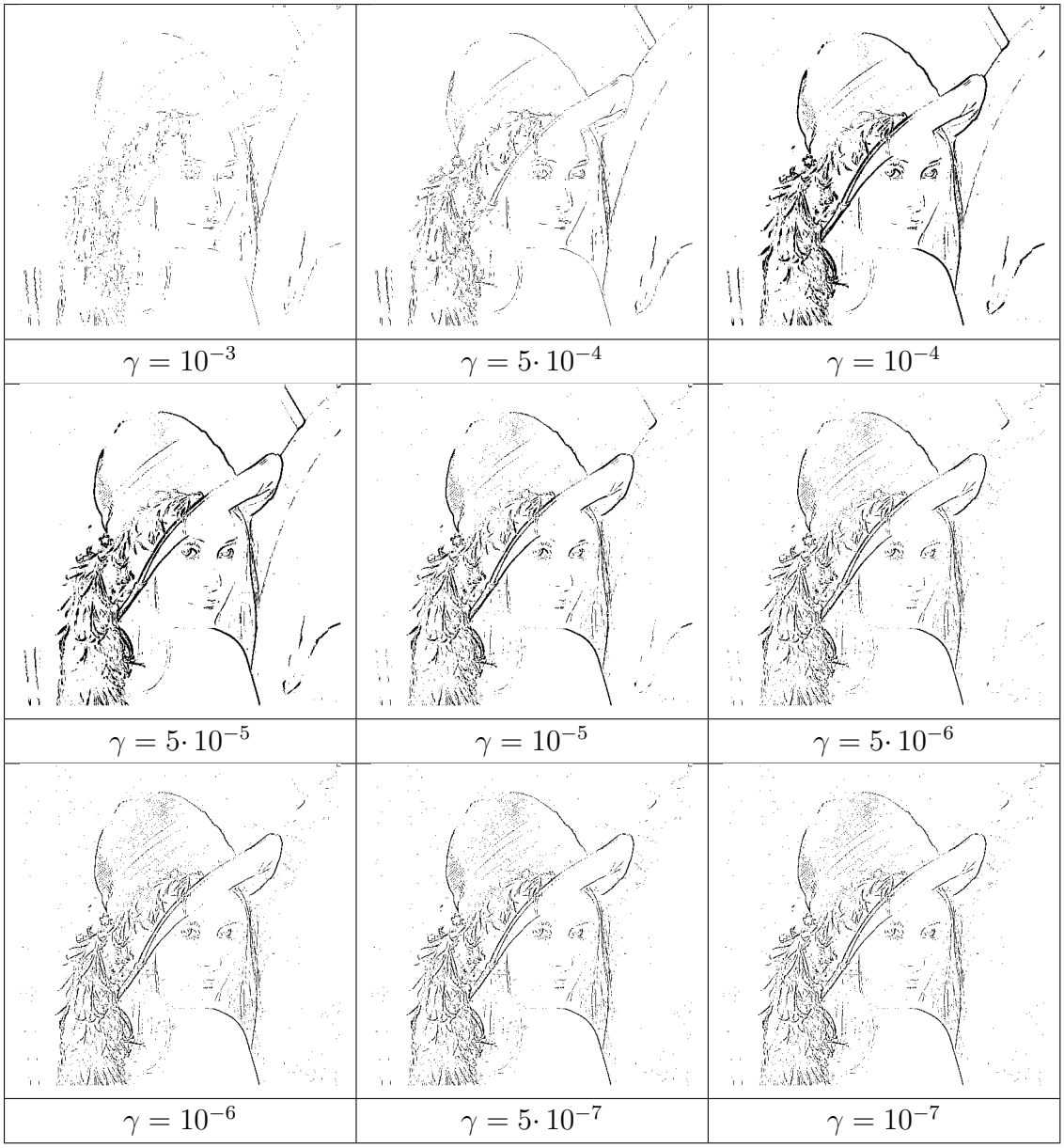
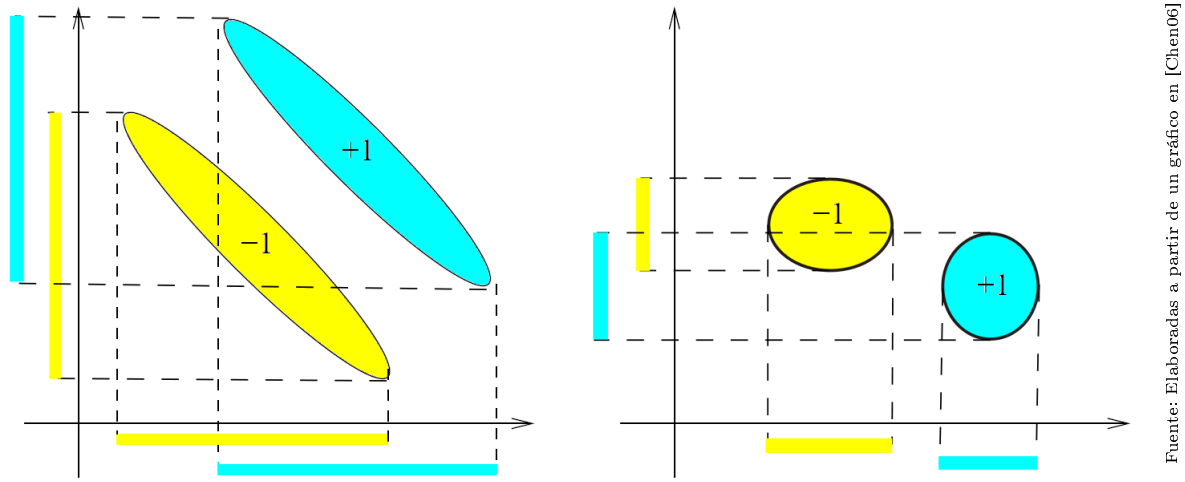



Figura 3.22: Pruebas realizadas usando el kernel gaussiano con distintos valores de $\gamma$.




**(a)** **(b)**

Figura 3.23: Ejemplos de conjuntos separables con información mutua y sin información mutua. (a) Ejemplo en el que dos conjuntos de patrones no mostrarían la información mutua entre características (ambos F-score serían bajos) aunque serían separables mediante una combinación de ambas, (b) Ejemplo de conjuntos separables en los que una de las características tendría F-score alto y la otra bajo

es la $i_{\text{ésima}}$ característica del $k_{\text{ésimo}}$ vector negativo.Cuanto mayor es el "F-score" más probable es que esta característica sea más discriminativa.

El numerador indica la discriminación entre los conjuntos positivo y negativo. El denominador indica la discriminación dentro de los propios conjuntos positivo y negativo. En el numerador, cuanto mayor sea la distancia entre la media total y las medias particulares para cada característica en los conjuntos positivo y negativo, más separados estarán los conjuntos entre sí. En el denominador, cuanto más agrupados estén los conjuntos positivo y negativo alrededor de sus medias respectivas (para una determinada característica) más fácil sería separarlos. Por tanto, interesa un valor alto para el numerador y cuanto más bajo posible en el denominador.

Una desventaja del "F-score" es que no revela información mutua entre características. Es decir, aunque los conjuntos sean separables usando una combinación de características, "F-score" no lo mostrará puesto que las trata por separador. En la figura 3.23(a) puede verse un ejemplo. En la figura 3.23(b) se puede ver un ejemplo en el que los F-score de las características sí que permiten determinar la que permite discriminar ambos conjuntos.

En la tabla 3.2 se presentan las puntuaciones obtenidas con este método por cada uno de los píxeles en una ventana $3 \times 3$ alrededor del píxel a clasificar. Se han marcado en negrita los valores que son más destacados en cada uno de los casos de detección que se han analizado. Para el caso de la detección de bordes en todas las direcciones, los píxeles que obtienen más puntación son 2, 4, 5, 6 y 8 que se presentan de manera gráfica en la figura 3.24(a). En el caso de sólo detección de bordes verticales aparecen como relevantes los píxeles 4, 5 y 6 (Figura 3.24(b)) y en el caso horizontal 2, 5 y 8 (Figura 3.24(c)). Estos resultados parecen acorde a la experiencia en la detección de bordes y al entrenamiento

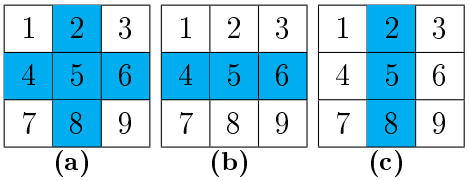



Figura 3.24: Presentación gráfica de los resultados obtenidos en la tabla 3.2. (a) Detección total, (b) Detección vertical, (c) Detección horizontal.

utilizado. En todo caso, la reducción en el número de características solo será relevante si dichos resultados se comprueban de manera gráfica en la detección de los bordes.

En la figura 3.25 se presentan los resultados obtenidos usando sólo los píxeles seleccionados. La figura 3.25(a) muestra la imagen correspondiente a la detección de bordes con los píxeles 2, 4, 5, 6 y 8, puede comprobarse como el resultado no es exactamente igual al presentado en la imagen 3.19(c) pero las diferencias son mínimas. Las imágenes en 3.25(b)-(c) muestran respectivamente los resultados para la detección de bordes horizontales usando los píxeles 2, 5 y 8 y los resultados para la detección de bordes verticales con los píxeles 4, 5 y 6. Nuevamente la comparación con resultados anteriores (Figura 3.20) muestra que la reducción de características no influye en gran manera en la detección de los bordes. Quizás pueda resaltarse en la imagen 3.25(b) como la parte inclinada de la flecha sí aparece marcada mientras que en los resultados anteriores no.

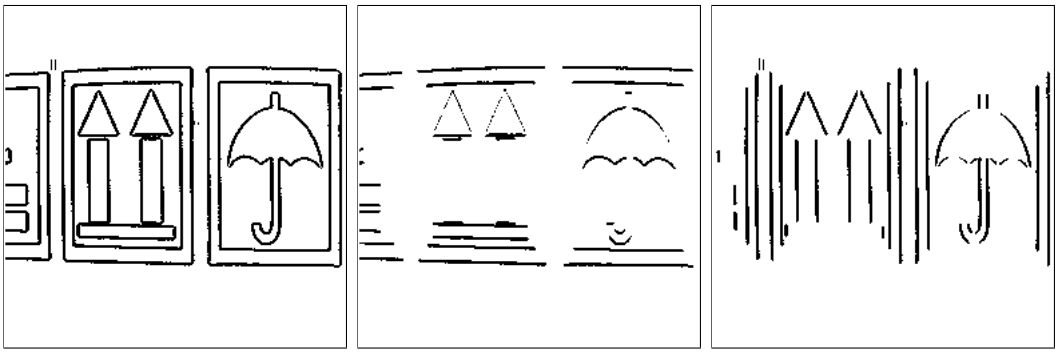



Figura 3.25: Comprobación de la detección de bordes utilizando las características seleccionadas. (a) Todos los bordes, (b) Bordes horizontales, (c) Bordes verticales.

Con las pruebas realizadas parece claro que una reducción en el número de píxeles utilizados para la detección de los bordes no merma significativamente los buenos resultados aunque sí que produce variaciones en los mismos. Estas variaciones son mínimas aunque podrían ser relevantes en algún caso. Por otro lado, la reducción de características mejora la rapidez de ejecución no sólo por la reducción en el tamaño de los vectores sino también porque en el entrenamiento se ha comprobado una reducción importante en el número de vectores soporte finales. En el caso de la detección total la reducción ha sido superior al 50 % y en la horizontal y vertical superior al 80 %.

**Tamaño de la ventana de entrenamiento.** Un parámetro a tener en cuenta en la generación del entrenamiento para la detección es el tamaño de la ventana elegida

Tabla 3.3: Comparación del número de vectores soporte para una ventana de $3 \times 3$ y una de $5 \times 5$. Diferentes valores de $\gamma$ y kernel gaussiano.

| $\gamma$ | $10^{-7}$ | $5 \cdot 10^{-7}$ | $10^{-6}$ | $5 \cdot 10^{-6}$ | $10^{-5}$ | $5 \cdot 10^{-5}$ | $10^{-4}$ | $5 \cdot 10^{-4}$ | $10^{-3}$ |
|---|---|---|---|---|---|---|---|---|---|
| #SV($3 \times 3$) | 39 | 15 | 18 | 16 | 22 | 75 | 115 | 420 | 639 |
| #SV($5 \times 5$) | 37 | 29 | 36 | 79 | 106 | 225 | 399 | 1285 | 1548 |

alrededor del píxel a clasificar. Por eso en este apartado se comparan resultados con distintos tamaños de ventana.

Para esta comparación se ha generado entrenamiento usando una ventana de $3 \times 3$ y una de $5 \times 5$ y se ha repetido el proceso de entrenamiento mostrado en la imagen 3.22 para la comparación de resultados con distintos valores de $\gamma$ y con kernel gaussiano. Para las pruebas se utilizaron imágenes de entrenamiento sintéticas con una diferencia de 50 y un nivel de ruido de 5.

En la figura 3.26 se pueden observar los efectos de usar la ventana de $5 \times 5$ comparando con lo presentado en la figura 3.22. De esa comparación se pueden obtener dos conclusiones principales. La primera es que los resultados óptimos (bordes más marcados) se dan a un valor de $\gamma$ distinto, ligeramente inferior. La segunda es que el uso de esta ventana mayor resta sensibilidad al detector, haciendo que muchos detalles desaparezcan.

Otro efecto importante es que el número de vectores soporte cambia en gran manera de una ventana a otra. Como se aprecia en la tabla 3.3 el número de vectores es mayor, en general, para la ventana más amplia y para los valores óptimos de $\gamma$ la diferencia va de 75 con $3 \times 3$ a 106 con $5 \times 5$.

**Detección mediante diferencias.** Durante todo este apartado se ha realizado un entrenamiento en el que se tomaban directamente los valores de gris alrededor de un píxel dado (incluyendo a éste). Sin embargo, puesto que los bordes marcan los saltos entre zonas con valores de gris distintos, puede ser interesante realizar el entrenamiento usando las diferencias entre el píxel central y los valores de gris a su alrededor. De esta forma lo importante no serían los valores absolutos sino las diferencias entre píxeles contiguos. El entrenamiento se puede simplificar si se utiliza esta opción puesto que el número de vectores posibles se reduce al ser importantes sólo las diferencias. Esto se reflejará en el número de vectores soporte generados puesto que se reducirán.

La primera prueba a realizar nos permitirá comparar los resultados obtenidos con y sin el uso de diferencias. Para ello repetimos la prueba de la figura 3.22 pero esta vez usando diferencias. Los resultados para distintos parámetros de entrenamiento se muestran en la figura 3.27. Una comparación entre ambas figuras nos da información de que los resultados son similares aunque el uso de diferencias hace que aparezcan más detalles y texturas.

Una diferencia muy importante entre el uso de diferencias o sólo de valores aparece cuando comparamos el número de vectores soporte obtenidos tras el entrenamiento. En la tabla 3.4 aparecen comparados los números de vectores en cada caso. La reducción (cómo se esperaba) es muy importante. Esa reducción implica una mayor rapidez en la ejecución del detector de bordes.

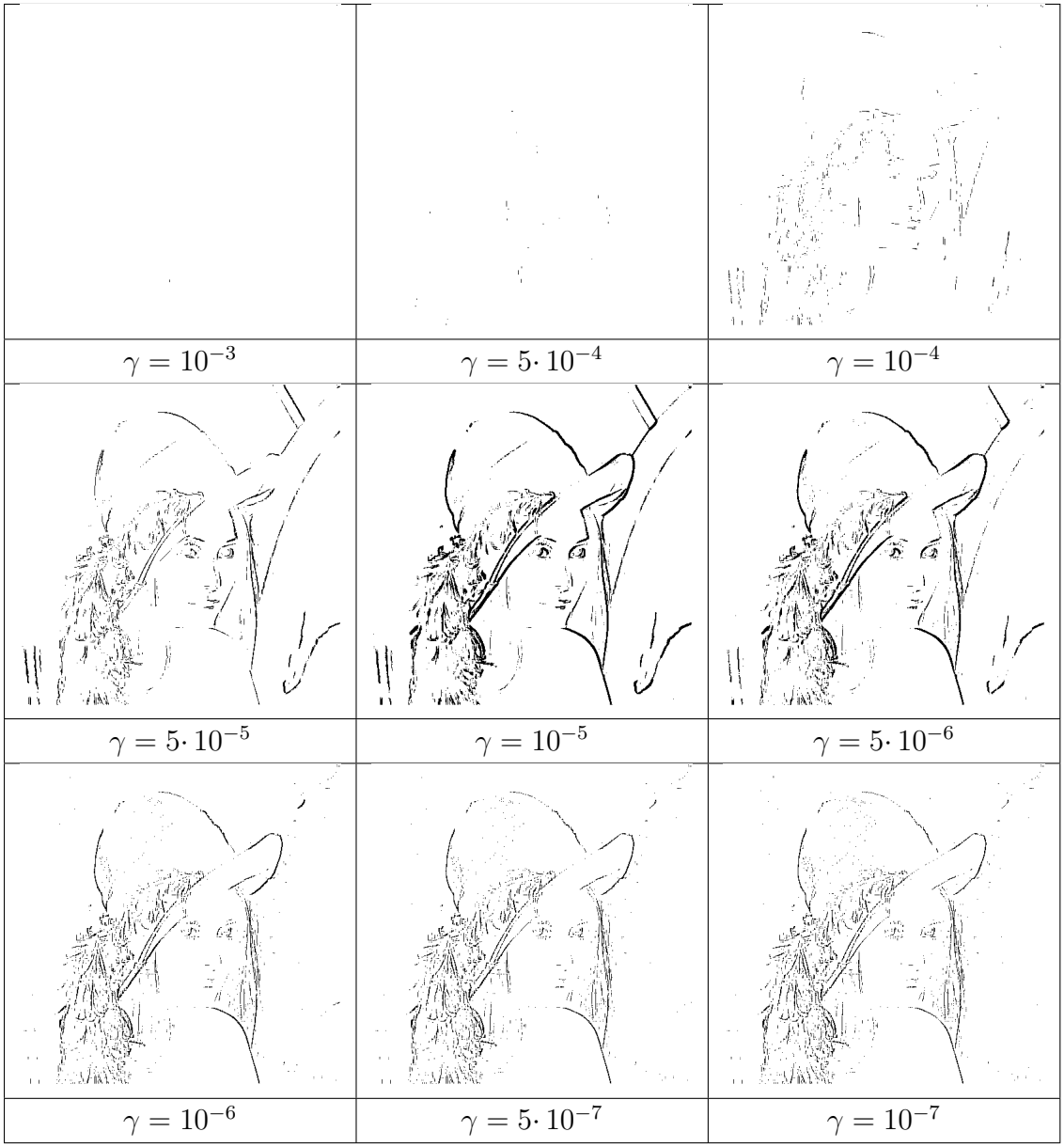



Figura 3.26: Pruebas realizadas usando el kernel gaussiano con distintos valores de $\gamma$. Ventana de $5 \times 5$.

Tabla 3.4: Comparación del número de vectores soporte para uso de valores y de diferencias. Diferentes valores de $\gamma$ y kernel gaussiano.

| $\gamma$ | $10^{-7}$ | $5 \cdot 10^{-7}$ | $10^{-6}$ | $5 \cdot 10^{-6}$ | $10^{-5}$ | $5 \cdot 10^{-5}$ | $10^{-4}$ | $5 \cdot 10^{-4}$ | $10^{-3}$ |
|---|---|---|---|---|---|---|---|---|---|
| #SV Valor | 39 | 15 | 18 | 16 | 22 | 75 | 115 | 420 | 639 |
| #SV Dif. | 14 | 6 | 5 | 5 | 5 | 6 | 11 | 25 | 28 |


Figura 3.27: Pruebas realizadas usando el kernel gaussiano con distintos valores de $\gamma$. Se usan diferencias en vez de valores.

### 3.2.2. Imagen de clasificación. Gradación en la detección de bordes.

En la sección anterior se han presentado resultados preliminares en los que se comprueba que, efectivamente, las máquinas de vectores soporte pueden ser un método efectivo para la detección de bordes. Sin embargo, los métodos presentados en el estado del arte daban como resultado unos bordes mucho más localizados, esto es, se seleccionaba como borde sólo un píxel de los posibles obtenidos mediante la aplicación del gradiente o del método elegido en cada momento. Para llevar a cabo esa selección existen varias posibilidades que van, desde la aplicación de operaciones morfológicas, a la búsqueda de máximos locales en la imagen de bordes. La técnica más generalizada en la literatura sobre el tema es la búsqueda de máximos locales, por ejemplo, en el detector de Canny ([Canny86]) ese paso se conoce como "non-max supression". Sin embargo, esta técnica necesita que los valores devueltos por el detector de bordes no sean 0 o 255 sino que haya una gradación en los resultados de la detección de manera que se pueda elegir, de los puntos detectados como borde, los que obtengan un valor mayor.

Para llevar a cabo dicha gradación se proponen en este punto dos técnicas distintas:

1. La función de clasificación $f(\mathbf{x})$ devuelve un valor distinto para cada vector de entrada y se considera que el vector pertenece a una clase si es positivo y a la otra si es negativo. Sin embargo, el valor devuelto por la función dependerá de lo alejado que se encuentre el vector de la frontera de decisión y de la zona en la que se encuentre (positiva o negativa). Por ello dicho valor nos dará una idea del grado de pertenencia a una determinada clase. Basándonos en esto podemos generar una imagen de bordes con distintos niveles de gris en según el valor devuelto por la función de clasificación sin más que realizar una normalización y un cambio de escala según la siguiente ecuación:

$$borde(\mathbf{x}) = \left(1 - \frac{f(\mathbf{x}) - f_{\min}}{f_{\max} - f_{\min}}\right) \cdot 255. \tag{3.9}$$

   Un ejemplo de esta técnica se puede ver en la imagen 3.28 dónde los píxeles próximos al negro corresponden a los valores máximos devueltos por la función de clasificación y los valores próximos al blanco a los valores mínimos. Entre ellos aparecen una gama de grises indicando la gradación en la detección de los bordes. En los detalles de las imágenes 3.28(c) y 3.28(d) se puede comprobar mejor esta gradación.

2. Otra posibilidad para marcar el grado de pertenencia a la clase borde sería la obtención de la probabilidad de pertenecer a dicha clase partiendo de la información obtenida en la clasificación. Para ello utilizaremos la técnica de Platt [Platt99] en la que se utiliza una función sigmoidal parametrizada para conseguir la probabilidad de pertenecer a una clase dado el valor devuelto por la función de clasificación. Dicha función sigmoidal tiene la siguiente expresión:

$$P(y = 1|f(\mathbf{x})) = \frac{1}{1 + e^{Af+B}}. \tag{3.10}$$

   Por tanto la ecuación de la que se obtendrá la imagen de bordes será:

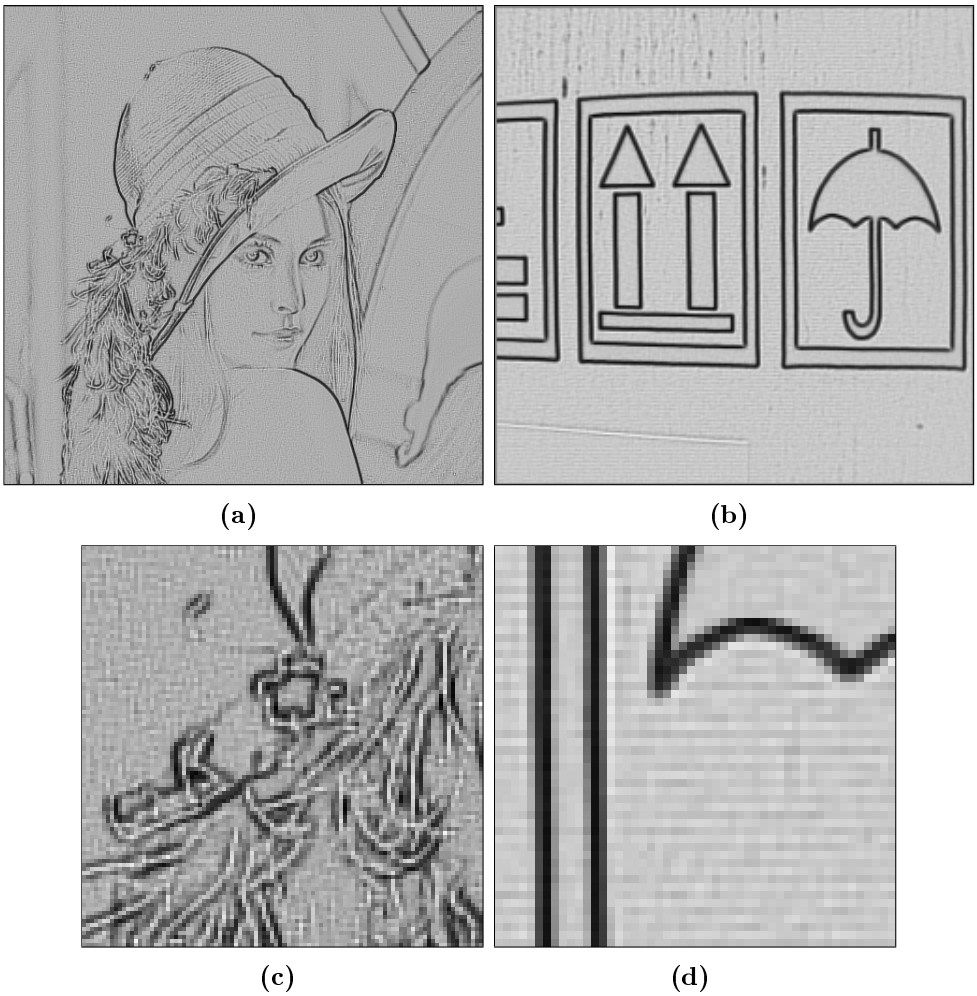



Figura 3.28: Obtención de una imagen de bordes mediante normalización y cambio de escala. (a) Ejemplo sobre la imagen Lenna; (b) Ejemplo sobre la imagen Pattern; (c) Detalle de la imagen (a); (d) Detalle de la imagen (b).

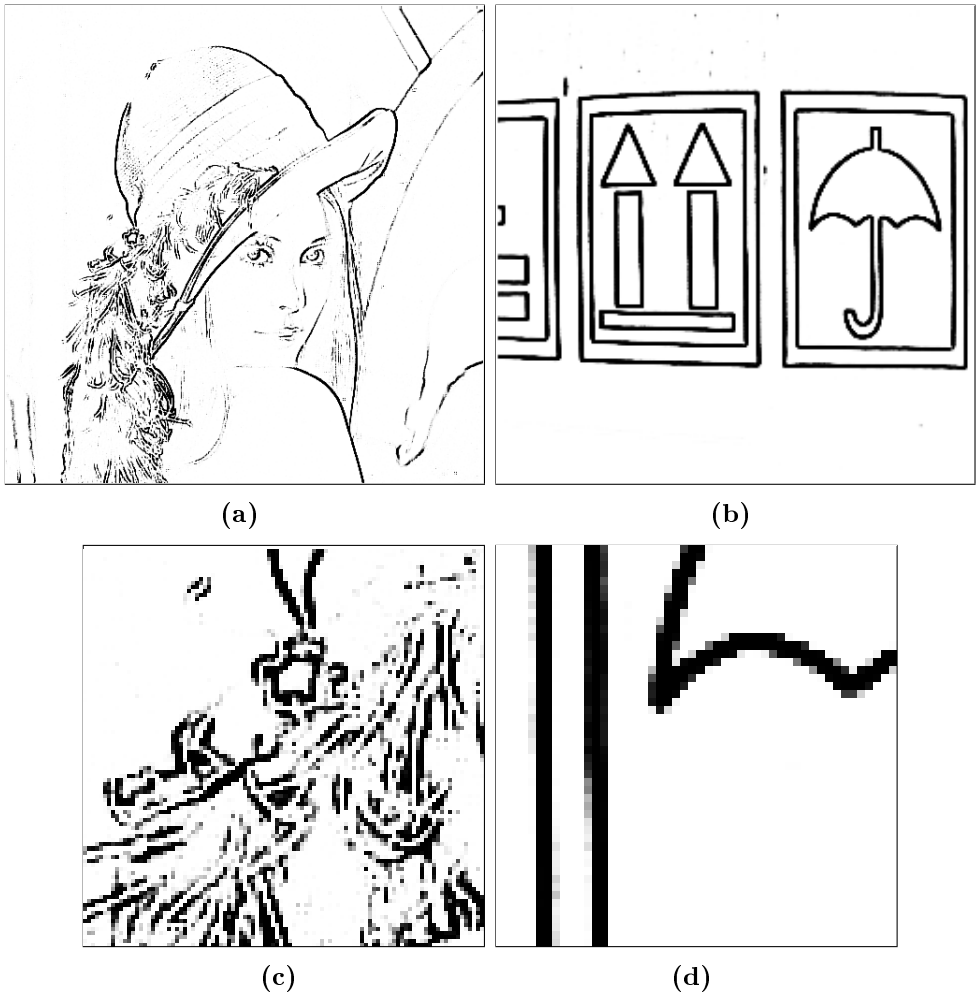



Figura 3.29: Obtención de una imagen de bordes mediante el uso de la función de probabilidad. (a) Ejemplo sobre la imagen Lenna; (b) Ejemplo sobre la imagen Pattern; (c) Detalle de la imagen (a); (d) Detalle de la imagen (b).

$$borde(\mathbf{x}) = (1 - P(y = 1|f(\mathbf{x}))) \cdot 255. \tag{3.11}$$

Un ejemplo de esta técnica se puede ver en la imagen 3.29 dónde los píxeles próximos al negro corresponden a los valores máximos de probabilidad y los valores próximos al blanco a los valores mínimos. Entre ellos aparecen una gama de grises indicando la gradación en la detección de los bordes. En los detalles de las imágenes 3.29(c) y 3.29(d) se puede comprobar mejor esta gradación.

Si se observan las imágenes de ejemplo para cada una de las técnicas propuestas aparecen diferencias debidas a que las formas de trasladar la información de clasificación a la detección de bordes son distintas. En el caso de la normalización y cambio de escala la transformación utilizada es lineal, y por lo tanto aparece una gama de grises más amplia indicando más gradación en la imagen de bordes. Sin embargo, la función de probabilidad no es lineal y tiende a saturar los valores próximos a los extremos y las imágenes obtenidos son menos ricas en valores de gris. La explicación se puede ver en la figura 3.30

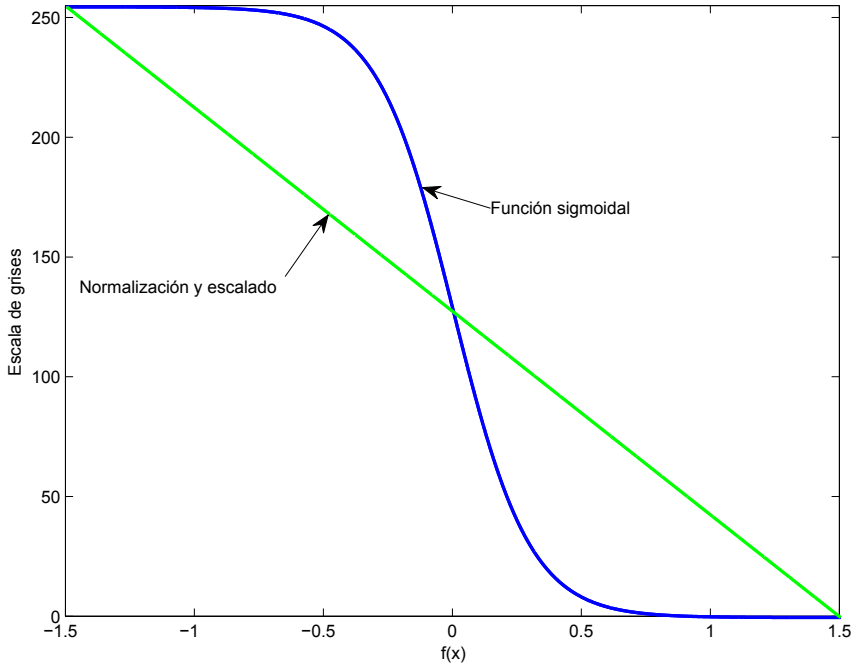



Figura 3.30: Comparación de las funciones de generación de escala de grises.

donde se observa la diferencia entre ambas funciones y donde se puede comprobar que la función sigmoidal alrededor de los valores máximos y mínimos se mantiene prácticamente constante, y por tanto no discrimina entre valores de gris en esa zona.

Ambas formas de obtener la gradación en los valores correspondientes a las imágenes de borde son igualmente válidas, aunque el uso de la función de probabilidad tiene un mejor respaldo según la teoría estadística y la técnica de normalización y escalado se basa más en una mera observación empírica.

### 3.2.3. Marcado de píxeles.

Como conclusión final, es necesario definir el método por el cual se obtendrán las imágenes correspondientes a los bordes de una imagen de una manera similar a las utilizadas en la literatura sobre el tema. Es decir, el resultado final del método propuesto debe ser una imagen en la que los bordes aparezcan marcados con una curva cuya anchura sea de un píxel y que defina la frontera de separación entre zonas de la imagen.

Después de un análisis de las posibilidades, se proponen en este apartado cuatro posibilidades:

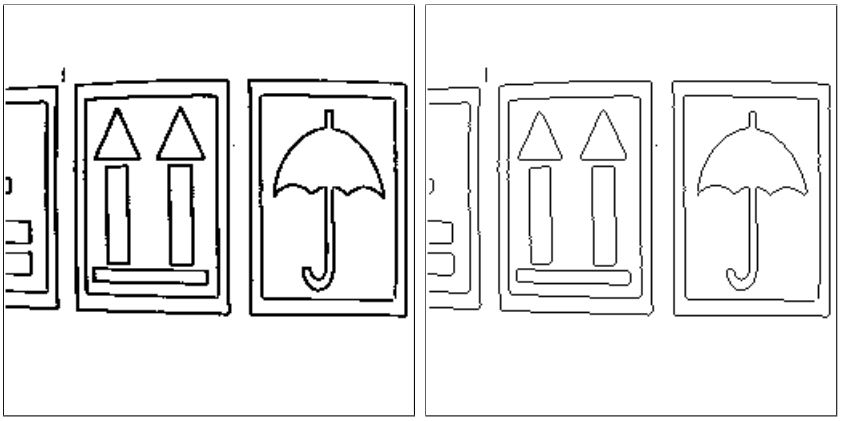
1. Utilizar un algoritmo de adelgazamiento ("Thinning" en inglés) sobre las imágenes de bordes previamente umbralizadas. De esta forma se puede usar un sólo modelo de entrenamiento y usar un único parámetro (aparte de los de entrenamiento) para la obtención de los bordes.

2. Buscar máximos en las imágenes de bordes. Similar al método de supresión de no máximos utilizado en el método de Canny pero más sencillo.

3. Basándonos en el método de Canny, se puede aplicar la supresión de no máximos e histéresis según la dirección de bordes en la que nos movamos. Para ello es necesario detectar por separado los bordes horizontales y verticales. Por tanto se necesitan dos modelos de entrenamiento.

4. Si se detectan por separado los bordes horizontales y verticales es posible quedarse con los máximos de cada detección y unirlos para formar una sola imagen. Este método es simple y rápido aunque necesite de dos modelos.

**Adelgazamiento.** En este apartado se propone usar para la detección de bordes un umbralizado de la imagen de gris obtenida con alguna de las técnicas propuestas en la sección 3.2.2 seguido de un algoritmo de adelgazamiento de los muchos propuestos en la literatura. Concretamente, en esta tesis se utilizará el propuesto en [Cychosz94]. Este es un método iterativo en el que, en sucesivas pasadas, se consiguen buenos resultados de adelgazamiento manteniendo las propiedades geométricas de los objetos obtenidos. Una ventaja adicional es que existe una versión en C de dicho algoritmo accesible para el que necesite usarla.



**(a)** **(b)**

Figura 3.31: Ejemplo del proceso de adelgazamiento. (a) Imagen de bordes obtenida umbralizando la probabilidad de bordes con Th=240, (b) Resultado del adelgazamiento después de tres pasadas.

En la figura 3.31 se puede ver un ejemplo de como se comporta este método. Después del umbralizado la imagen obtenida no marca los bordes con una línea de un píxel de grosor sino que más bien se marca una zona en la que existe más probabilidad de encontrar un borde. El algoritmo de adelgazamiento permite la obtención de la línea que marca el borde a partir de la imagen de umbralizado anterior. Como se puede ver en la imagen 3.31(b) el resultado se ajusta a los bordes esperados para la imagen. Obviamente este resultado final depende en gran medida del umbralizado anterior. La figura 3.32 muestra distintos resultados para dos imágenes distintas y dos métodos de obtención de la imagen umbralizada distintos (Lineal o probabilística). Puede comprobarse que existen diferencias entre los resultados pero estas son mínimas.

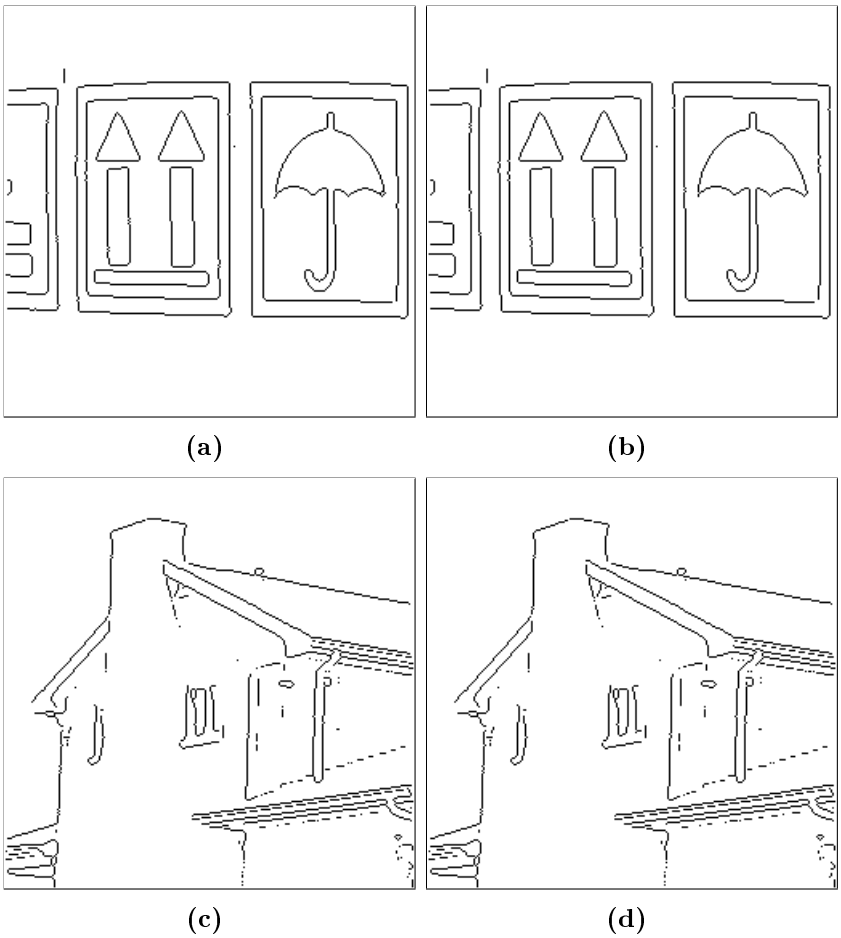
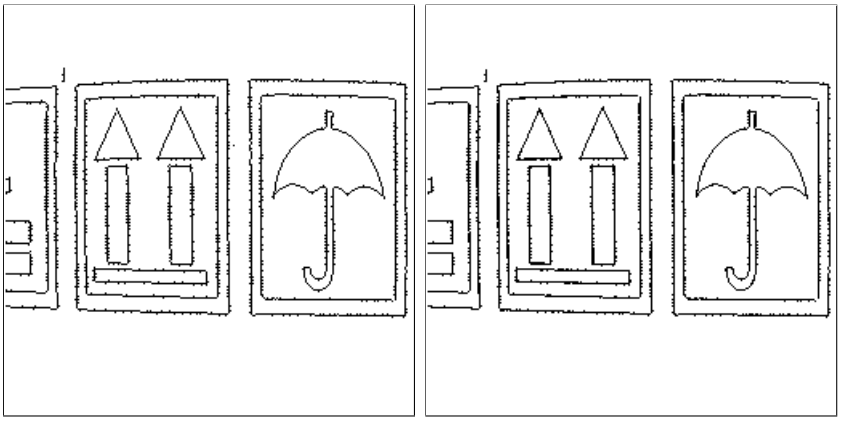


Figura 3.32: Aplicación del algoritmo de adelgazamiento. (a)-(c) Imágenes usando el método lineal, (b)-(d) Imágenes usando el método probabilístico.



(a) (b)

Figura 3.33: Efecto producido por la búsqueda de máximos directamente en el gradiente. (a) Gradiente lineal; (b) Gradiente de probabilidad.

**Búsqueda de máximos en la imagen de bordes.** Puesto que las imágenes de detección que se están obteniendo dan una gradación de la pertenencia de un píxel a un borde, se puede utilizar dicha información para marcar sólo como borde los píxeles que mayor valor presenten. Si se buscan los máximos directamente sobre la imagen de detección el resultado es que aparecen demasiados detalles, por tanto, esta búsqueda se realizará sólo sobre los píxeles que presenten un valor superior a un umbral dado. El marcado entonces es sencillo, se compara un píxel con el anterior y el posterior y si es mayor o igual que ambos se marca como "borde" y si no como "no borde". Esto se hace tanto en dirección vertical como horizontal. Sin embargo la imagen resultante presenta un aspecto como el de la figura 3.33. Puede observarse que tanto en los bordes verticales como en los horizontales aparecen pequeños segmentos parásitos provenientes de la detección en una dirección distinta a la del borde original.

Para la eliminación de ese efecto se barajaron distintas posibilidades, incluyendo operaciones morfológicas o algoritmos similares al adelgazamiento usado en la sección anterior. Todos ellos pasan por definir las relaciones que deben existir entre los píxeles de una determinada ventana para que su píxel central sea considerado como borde o como ruido. Para definir dichas relaciones hay que estudiar los píxeles parásitos y buscar las relaciones existentes con su entorno y así poder identificarlos y borrarlos. Un posibilidad, utilizada en algoritmos de adelgazamiento, es definir una batería de comparaciones para identificar los píxeles. Aunque esta es una manera eficiente, cada redefinición del problema supone una redefinición de las comparaciones.

En esta tesis se propone usar las SVM como clasificador de píxeles de manera empírica a modo de operaciones morfológicas. Para ello sólo hay que crear los vectores de entrenamiento de manera empírica y entrenar una SVM que clasifique los píxeles a borrar o a mantener. Por ejemplo, la figura 3.34 muestra diversas configuraciones de píxeles donde, en el caso de las imágenes de la fila superior el píxel central debe ser eliminado y en el caso de la fila inferior el píxel debe ser dejado como está. Cada una de esas configuraciones dará lugar a un vector de 9 componentes que entrenará a la SVM.

En el caso que nos ocupa se seleccionaron 32 configuraciones de píxeles de las cuales 10 estaban marcadas para borrar y el resto para permanecer sin cambio. Se entrenó una

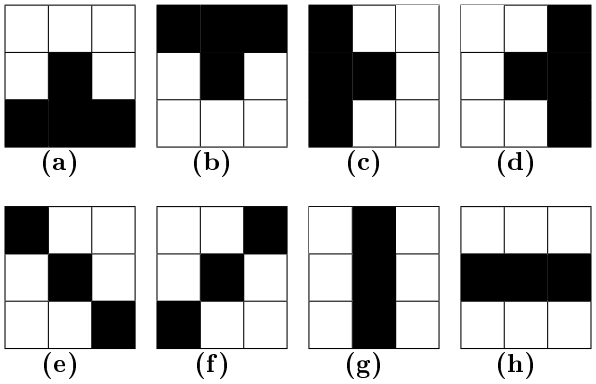



Figura 3.34: Ejemplo de definición de entornos de píxeles. (a)-(d) Entornos en los que el píxel central ha de eliminarse, (e)-(h) Entornos en los que el píxel central ha de mantenerse.

SVM con kernel gaussiano y una $\gamma = 10^{-5}$ dando como resultado un número de vectores soporte de 26. Después esta SVM se utilizó para postprocesar las imágenes de la figura 3.33 hasta conseguirse el resultado mostrado en las figuras 3.35(a)-(b). En las imágenes 3.35(c)-(d) se puede apreciar un ejemplo de la aplicación del proceso sobre una imagen real y observar que los resultados siguen siendo óptimos.

El procesado morfológico con SVM se muestra como una herramienta cómoda y rápida para procesar imágenes a nivel de píxel sin necesidad de realizar comparaciones explícitas. Aunque las pruebas se han realizado en un entorno $3 \times 3$ del píxel, se puede ampliar el entorno para incluir más posibilidades en los alrededores y que su procesado sea más acorde al resultado buscado, sobre todo en las esquinas y con bordes más complejos. También podría utilizarse este tipo de procesado para cerrar bordes que se han quedado abiertos y sin continuidad en las esquinas o en puntos significativos.

**Supresión de no máximos e histéresis.** Otra forma de llegar a la imagen final de bordes es la utilizada en el método de Canny (ver sección 3.1.4) que incluye, tras la obtención del gradiente, dos pasos fundamentales que son la supresión de máximos (nonmax supression) y la histéresis. En el primero se buscan los máximos en las imágenes de gradiente correspondientes a los bordes verticales y horizontales y se marcan como posibles bordes. En la segunda se comprueba si dichos posibles bordes están dentro de unos umbrales establecidos y si pertenecen a algún borde en una dirección determinada, si es así, se marcan como tales.

Para utilizar dicho método, en el caso que nos ocupa, simplemente se necesitan obtener los bordes en las dos direcciones, vertical y horizontal, y después fijar los parámetros para el marcado de los bordes ($T_{high}$, $T_{low}$). La obtención de los bordes horizontales y verticales se realiza según se explicó en la sección 3.2.1 mediante el entrenamiento adecuado y la obtención de dos modelos de clasificación que no tienen porqué utilizar los mismos parámetros de entrenamiento. Dado que estas imágenes producen demasiados detalles se añade un parámetro de umbralizado para usar sólo los valores por encima del mismo.

En la figura 3.36 se muestra un ejemplo de dicho proceso. La fila superior muestra las imágenes de gradiente lineal y la fila inferior las de probabilidad. Puede comprobarse que los resultados son muy similares.

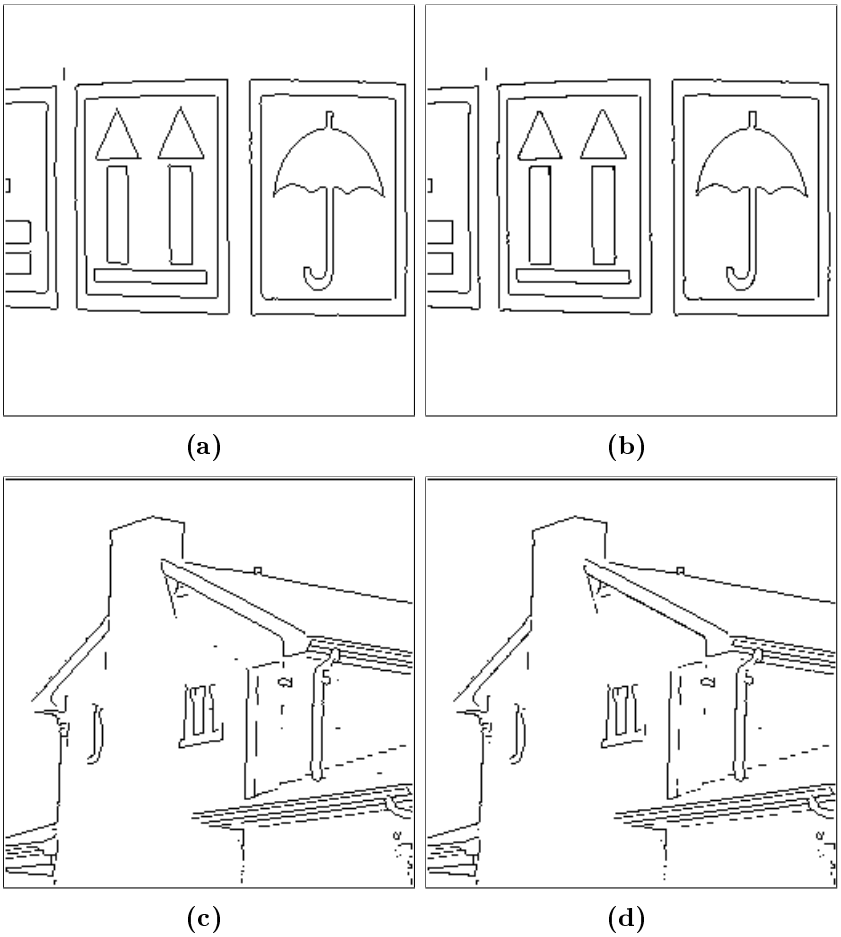



Figura 3.35: Resultado final tras la búsqueda de máximos y el procesado morfológico con SVM. (a) y (c) Usando gradiente lineal; (b) y (d) Usando gradiente de probabilidad.

En la figura 3.37 se muestra un ejemplo de resultado para este método pero aplicado para una imagen en la que no predominen figuras geométricas y se acerque más a las imágenes reales.

Esta forma de marcar los bordes es una posibilidad más, aunque en la práctica es difícil encontrar la combinación de parámetros más adecuada para encontrar los bordes correctos. Un incorrecta combinación de umbrales o de parámetros de entrenamiento produce la pérdida de algunos segmentos de borde.

**Búsqueda de máximos en horizontal y vertical.** El método presentado anteriormente busca los máximos en las direcciones ortogonales, horizontal y vertical, pero además añade dos direcciones más, 45 y 135 grados (oblicuos), que se pueden obtener de las anteriores. En un intento de simplificar la búsqueda de los bordes, en este punto se propone la búsqueda de máximos sólo en las direcciones vertical y horizontal y la composición directa de las imágenes resultantes para obtener el resultado final.

Ya en la página 58 dentro del punto 3.2.3 se propuso la búsqueda de máximos pero sobre la imagen completa de bordes. Este método funcionaba correctamente pero producía un efecto indeseado (ruido) que se eliminaba con un postprocesado morfológico basado

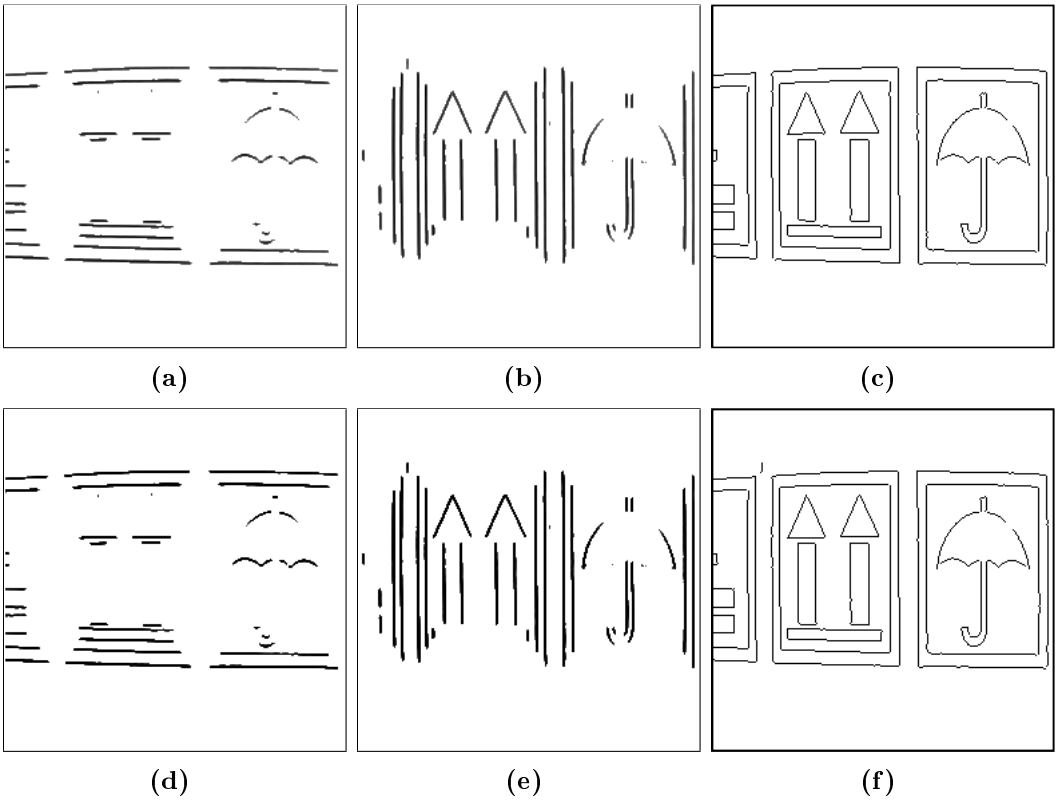



Figura 3.36: Ejemplos de obtención de bordes usando supresión de no máximos e histéresis. (a),(d) Bordes horizontales; (b),(e) Bordes verticales; (c),(f) Imágenes de bordes finales. En las imágenes (a)-(c) se utiliza gradiente lineal y en las (d)-(f) gradiente de probabilidad. En ambos casos Umbralizado=130, Thlow=0, Thhigh=0,2.

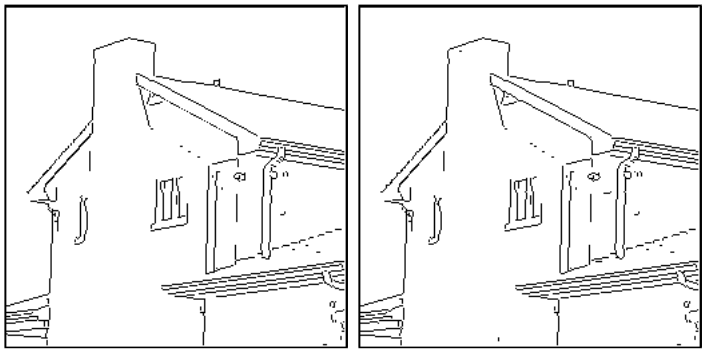
en SVM. La variación propuesta aquí hace innecesario el uso de ese postprocesado puesto que los bordes obtenidos no sufren de ese ruido asociado.

En la figura 3.38 se puede ver un ejemplo en el que se muestra directamente la composición final de los máximos horizontal y vertical en las dos imágenes de prueba que se han ido usando en este punto. Puede comprobarse como los resultados son comparables a los obtenidos mediante supresión de máximos e histéresis aunque habiendo simplificado el proceso.

La principal ventaja de este sistema es su simplicidad, puesto que sólo se necesita la búsqueda de máximos en dos direcciones y no hay postprocesado aunque para la obtención de las imágenes de borde se necesiten dos modelos, uno horizontal y otro vertical.

### 3.2.4. Comportamiento del método propuesto en presencia de ruido

Como se comentó en el apartado 3.2.1 las imágenes de entrenamiento llevan un pequeño nivel de ruido añadido para, por un lado, simular mejor los bordes que pueden aparecer en imágenes reales y obtener con ello un mejor entrenamiento, y por otro lado intentar que el método de detección tenga un modo de reducir el efecto del ruido presente en las



(a) (b)

Figura 3.37: Ejemplo de obtención de bordes mediante supresión de no máximos e histéresis en una imagen no geométrica. (a) Usando gradiente lineal con parámetros Umbralizado=130, Thlow=0, Thhigh=0,1. (b) Usando gradiente de probabilidad con parámetros Umbralizado=8, Thlow=0, Thhigh=0,1.

imágenes a la hora de detectar los bordes. Esta característica es deseable en los distintos métodos de detección de bordes, como ya se vio en el estado del arte.

En este punto se pretende evaluar si, realmente, el usar distintos niveles de ruido en el entrenamiento afecta a la detección de bordes en presencia de ruido en las imágenes. Para ello se realizaron distintas pruebas en las que se utilizaron distintos niveles de ruido para el entrenamiento. La imagen 3.39 muestra algunas imágenes de entrenamiento con distintos niveles de ruido. El nivel de ruido se entiende como los niveles de gris en los que pueden fluctuar los píxeles. Hay que resaltar que el ruido afecta de igual manera tanto a los píxeles de fondo como a los que pertenecen a un borde, de esta forma se pretende que la detección se vea afectada lo menos posible por la presencia de ruido puesto que pequeñas diferencias no son tratadas como borde en el entrenamiento.

Para realizar las pruebas se usaron imágenes a las que se les añadió un nivel de ruido moderado pero que se percibe claramente y puede afectar al detector. Las imágenes de la figura 3.40 son las que se utilizaron para dichas pruebas. Tienen un ruido gaussiano superpuesto de valor 10.

Llegados a este punto se pueden usar varias de las posibilidades propuestas anteriormente para la detección de los bordes y generar el entrenamiento. Se habrá de elegir:

- El entrenamiento (sobre todo la diferencia entre zonas claras y oscuras).
- El método de detección (por máximos, usando supresión de no máximos, con gradiente total o con gradiente vertical y horizontal).
- La forma de obtención del gradiente (lineal o probabilístico).

En la figura 3.41 se muestran resultados en los que se han utilizado imágenes de entrenamiento con una diferencia entre la zona clara y oscura de 30 (esto supone más sensibilidad que si se usan valores mayores) además se ha utilizado el método de supresión de no máximos (por tanto se han obtenido gradientes horizontales y verticales) y los gradientes se han obtenido linealmente, el resto de parámetros como el umbral de detección o los umbrales de supresión de no máximos han sido seleccionados para obtener los mejores

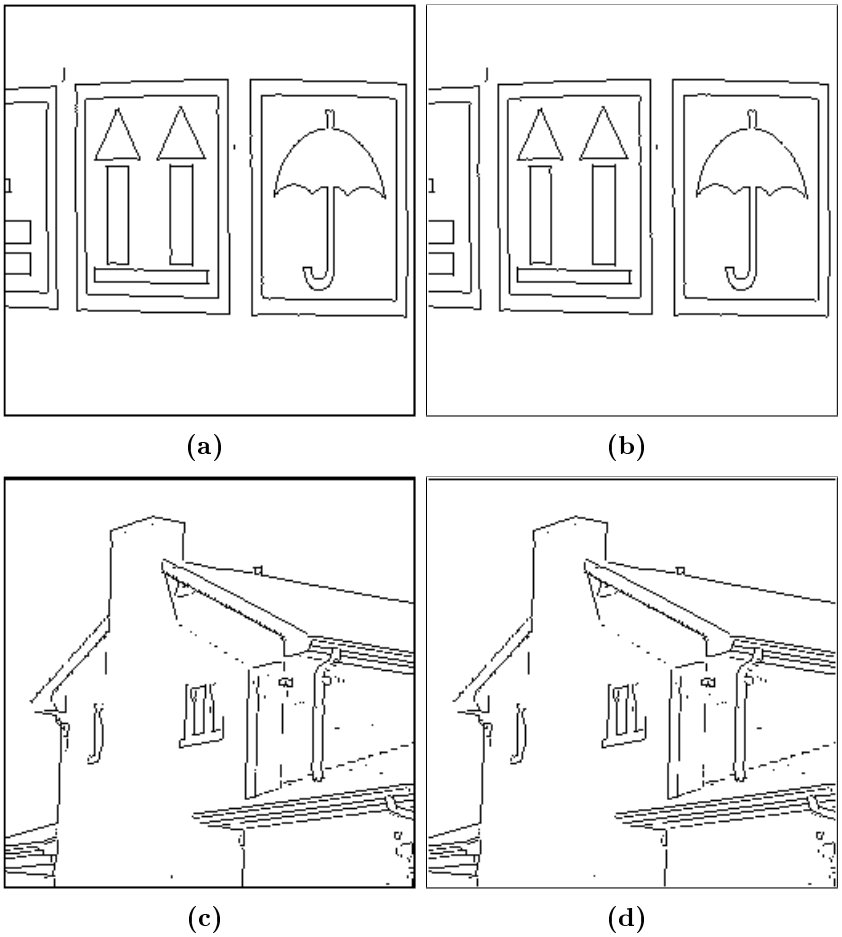
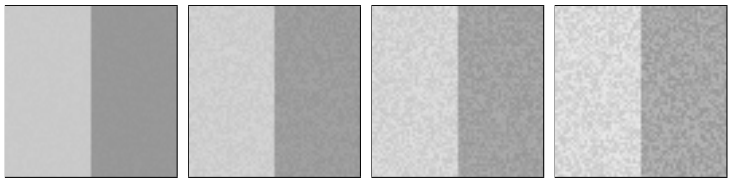



Figura 3.38: Ejemplo de búsqueda directa de máximos en gradientes vertical y horizontal. (a) y (c) Usando gradiente lineal; (b) y (d) Usando gradiente de probabilidad.

resultados. El detalle más interesante es ver como evoluciona la detección cuando el ruido de entrenamiento aumenta. En este caso se parte de un ruido 0 y en saltos de 5 se llega hasta 25, un valor superior ya pasaría por encima de la diferencia entre la zona clara y la oscura. Se puede comprobar que un mayor ruido en el entrenamiento "insensibiliza" el detector en cuanto al ruido aunque se mantienen en gran medida los resultados de detección.

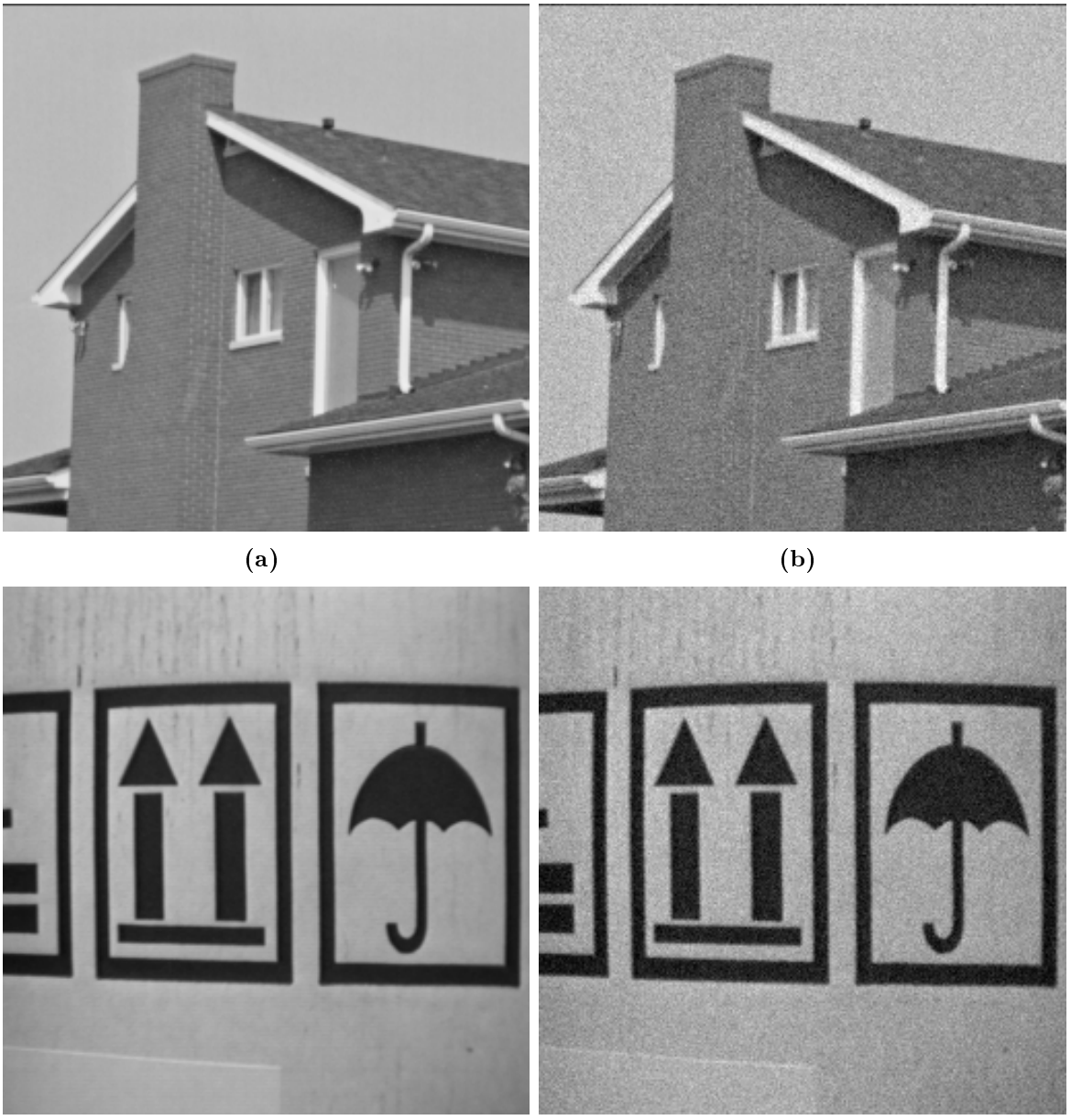
Otra manera de "insensibilizar" el detector es aumentando el valor de diferencia entre la zona clara y la oscura. Si aumentamos dicho valor y seguimos añadiendo ruido al entrenamiento se consiguen dos mejoras, por un lado el ruido influye menos en la detección y por otro aumentamos el nivel de ruido que se puede añadir al entrenamiento. En la figura 3.42 puede verse el resultado de aumentar la diferencia entre zonas de 30 a 50. Con ello se puede aumentar el ruido de entrenamiento hasta 40 y las imágenes obtenidas tienen menos ruido y los bordes son más definidos que con una diferencia de 30. Obviamente, esto sucede a costa de perder sensibilidad.

Un detalle interesante es comprobar como la adición de ruido al entrenamiento no aumenta significativamente el número de vectores soporte. Podría pensarse que, al añadir ruido, el número de posibles vectores que queden en la zona fronteriza aumenta ya que



(a) (b) (c) (d)

Figura 3.39: Ejemplos de imágenes de entrenamiento con distintos niveles de ruido. (a) Amplitud 5; (b) Amplitud 15; (c) Amplitud 25; (d) Amplitud 35.



(a) (b)

(c) (d)

Figura 3.40: Ejemplos de imágenes con ruido gaussiano. (a) y (c) Imágenes originales; (b) y (d) Imágenes con ruido añadido.

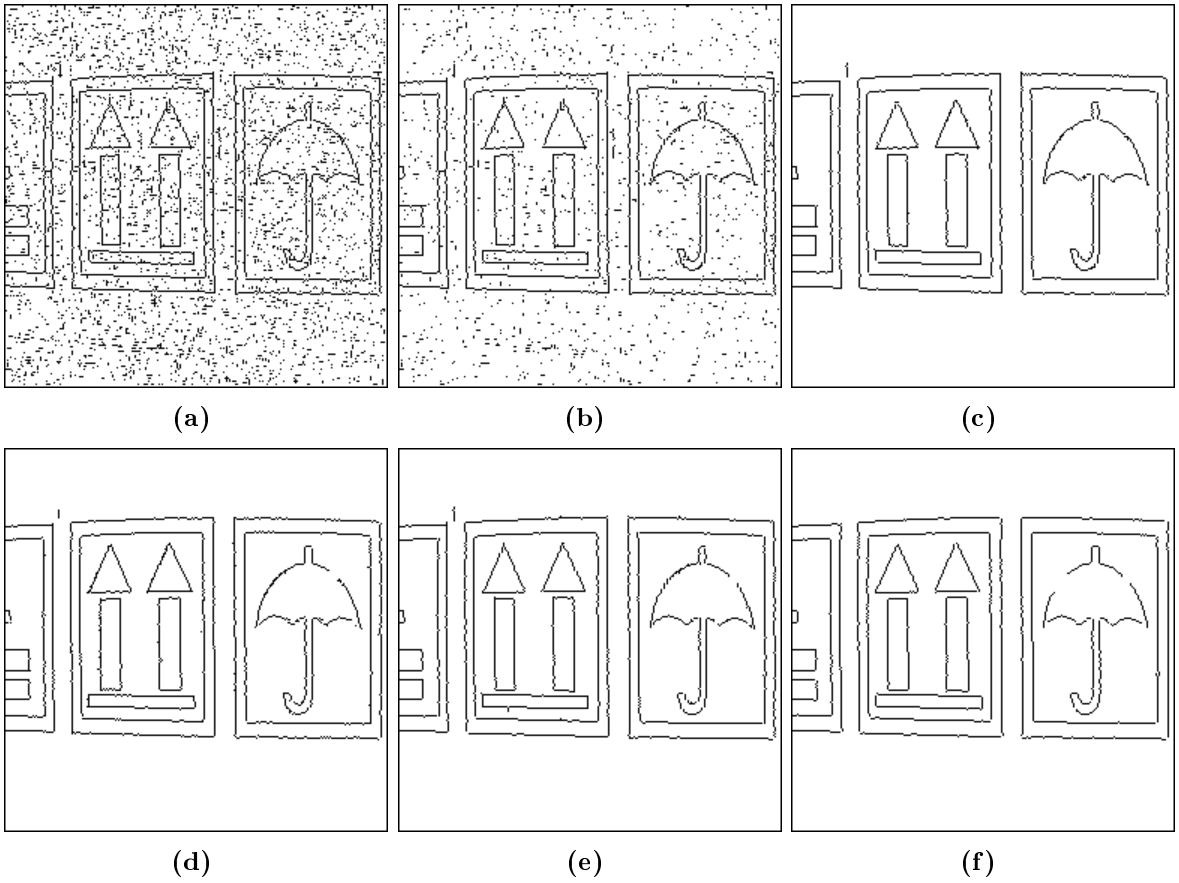



Figura 3.41: Ejemplo de obtención de bordes en presencia de ruido con una diferencia entre zonas de 30. Se presentan resultados para distintos niveles de ruido en el entrenamiento: (a) Nivel 0; (b) Nivel 5; (c) Nivel 10; (d) Nivel 15; (e) Nivel 20; (f) Nivel 25

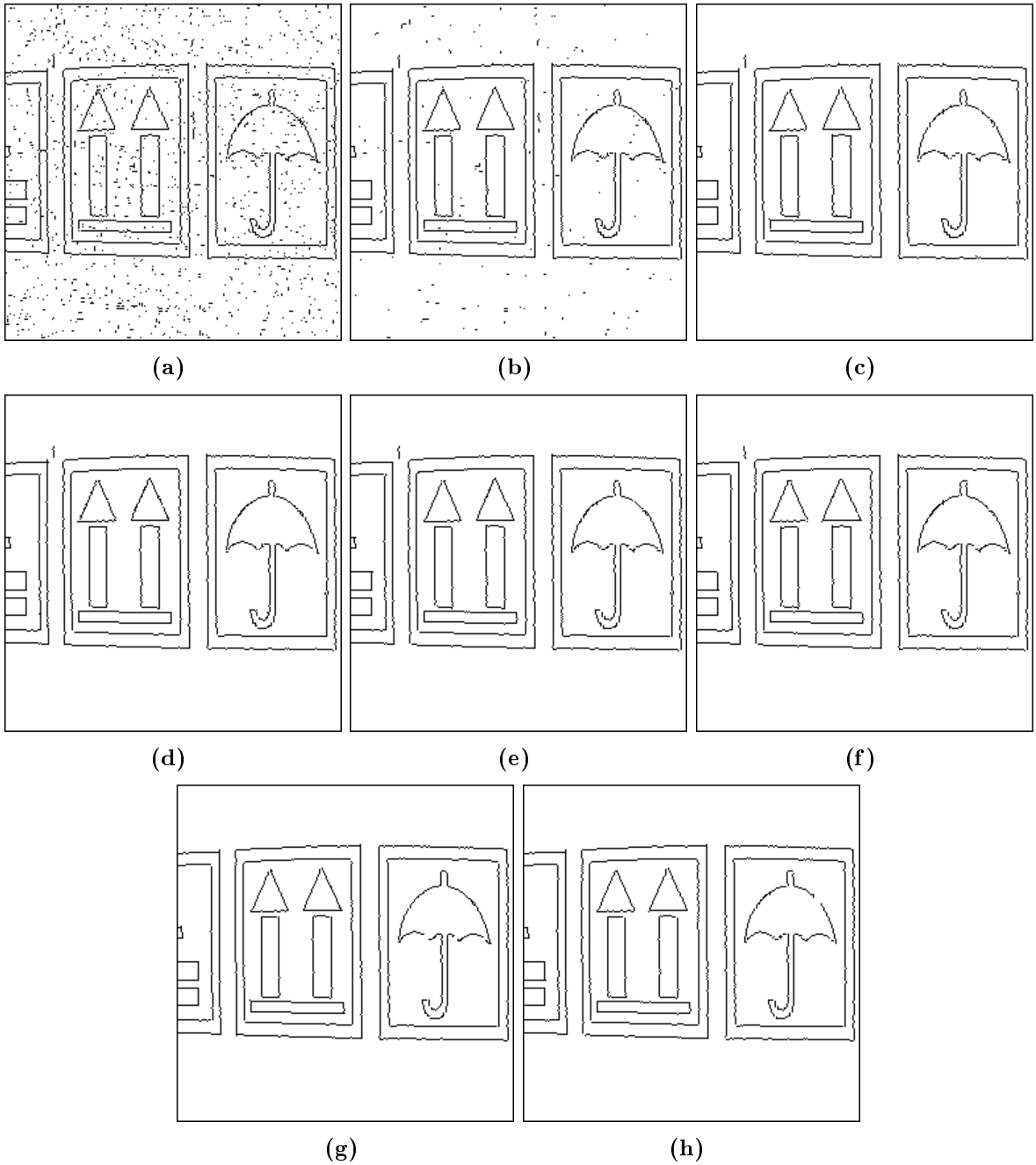



Figura 3.42: Ejemplo de obtención de bordes en presencia de ruido con una diferencia entre zonas de 50. Se presentan resultados para distintos niveles de ruido en el entrenamiento: (a) Nivel 0; (b) Nivel 5; (c) Nivel 10; (d) Nivel 15; (e) Nivel 20; (f) Nivel 25; (g) Nivel 30; (h) Nivel 35.

Tabla 3.5: Número de vectores soporte en función del ruido de entrenamiento. $\gamma = 5 \cdot 10^{-5}$ y diferencia entre zonas de 50.

| Amplitud de ruido | 0 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 |
|---|---|---|---|---|---|---|---|---|---|
| # SV Horizontal | 77 | 74 | 71 | 105 | 114 | 127 | 118 | 122 | 110 |
| # SV Vertical | 73 | 75 | 84 | 80 | 92 | 107 | 101 | 102 | 108 |

hay más vectores conflictivos que clasificar e información que se ha de incorporar al entrenamiento. Sin embargo, como puede verse en la Tabla 3.5 el número de vectores sí que aumenta ligeramente con el ruido pero se mantiene en unos valores muy próximos al caso en que el ruido es pequeño o no está presente. Este hecho influye en que la velocidad del detector de bordes no se ve claramente afectada por el ruido del entrenamiento aunque, como se ha visto, sí que mejora la calidad de detección.

Una vez que se ha comprobado que el ruido añadido al entrenamiento reduce la presencia del mismo en las imágenes de bordes, se pretende observar el efecto del nuevo entrenamiento y del ruido en los distintos métodos propuestos. Concretamente se compararán los métodos de gradiente total con búsqueda de máximos, gradiente total con adelgazamiento, gradiente vertical y horizontal con búsqueda de máximos y gradiente horizontal y vertical con supresión de no máximos. En todos los casos se usará la obtención lineal del gradiente porque la probabilística, debido a su mayor sensibilidad, empeora los resultados obtenidos y dificulta la búsqueda de los mejores resultados. En la figura 3.43 se pueden comparar visualmente los resultados obtenidos. Los parámetros para el entrenamiento utilizados en los distintos métodos son los mismos, siendo el ruido de entrenamiento de 35 y la diferencia entre zonas de 50. De la comparación se puede obtener la conclusión de que los resultados son similares, si bien el método de gradiente total obtiene más detalles. No hay, por tanto, un método superior a los demás en resultados ya que todos ellos permiten la reducción del efecto del ruido en la detección.

## 3.3. Comparación de métodos

En este apartado se presentará una comparación entre los métodos de segmentación presentados basada en medidas objetivas de calidad. Esta no es una tarea fácil como se verá también en el capítulo dedicado a la segmentación en color. Son muchos los intentos por definir medidas de calidad objetivas para la detección de bordes (en [Bowyer98] se puede encontrar una recopilación de los mismos) aunque ninguna satisface completamente el criterio de objetividad. Hay intentos de definir bancos de imágenes sintéticas que puedan servir como base para la medida de calidad ([Abdou79]) pero tienden a escoger aquellas características que benefician a un determinado algoritmo. Hay también intentos de mezclar medidas objetivas y subjetivas como el trabajo de Heath en [Heath97] pero cuya infraestructura no permite generalizarlos. Queda claro entonces la subjetividad implícita en la detección de bordes y lo difícil que resulta encontrar medidas objetivas de calidad.

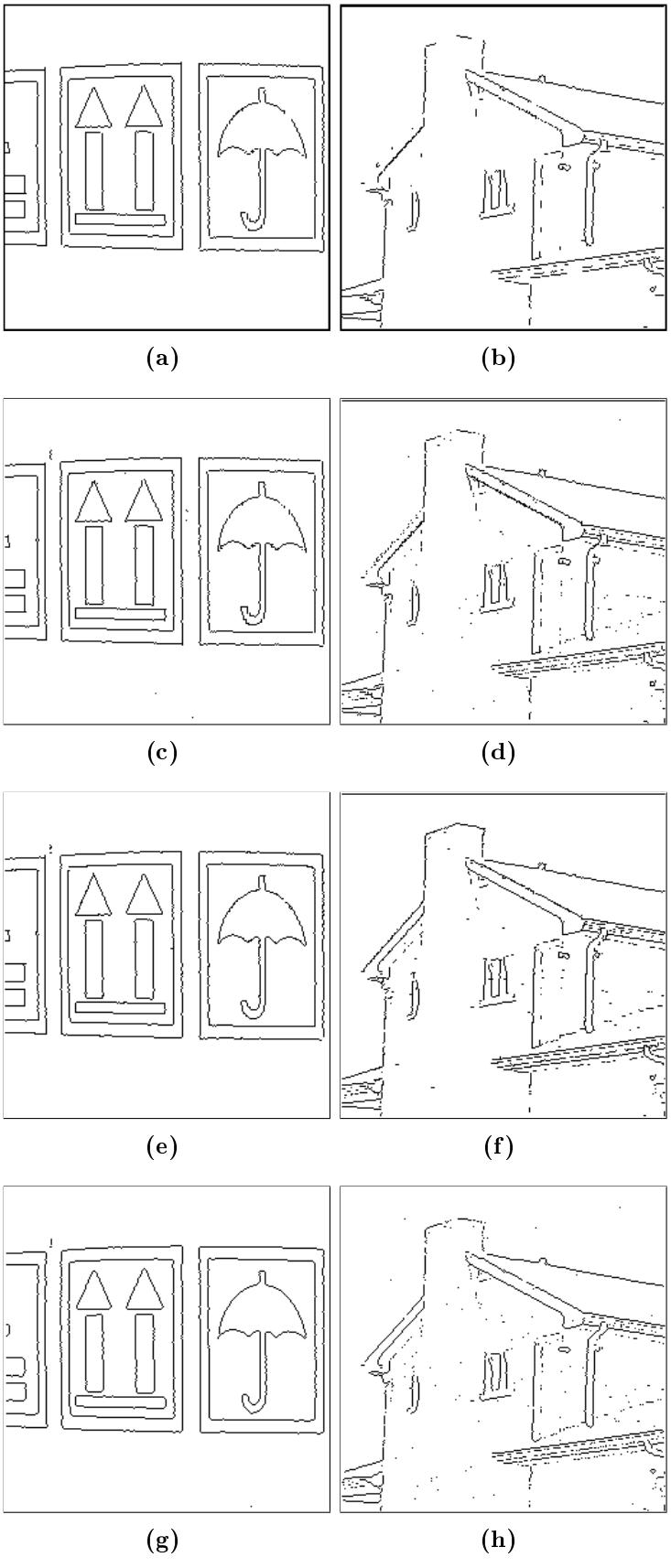



Figura 3.43: Comparación de distintos métodos en presencia de ruido. (a),(b) Supresión de no máximos; (c),(d) Búsqueda de máximos; (e),(f) Gradiente total con búsqueda de máximos; (g),(h) Gradiente total con adelgazamiento. En todos los casos el ruido de entrenamiento es de 35 y la diferencia entre zonas de 50.

Por ello, en esta tesis se propone el uso de una medida de calidad (Pratt's Figure of Merit, FOM) basada en imágenes de "ground-truth" que es sencilla de aplicar y que sí permite comparar algoritmos y evidenciar si alguno de ellos no obtiene la calidad adecuada. Además, hay estudios comparativos de métodos de evaluación para detección de bordes, por ejemplo [Chabrier04a], que demuestran que el índice FOM es el método más efectivo de los propuestos hasta el momento.

A esta medida se añade otra que nos permitirá conocer el grado de sobresegmentación ya que la anterior tiende a valorar mejor las imágenes sobresegmentadas.

### 3.3.1. Medidas de calidad

**Pratt's Figure of Merit.** La medida de calidad de detección de bordes conocida como "Figure of merit" [Pratt01] fue propuesta por Pratt para medir la exactitud de los bordes detectados en una imagen. Esta medida presenta la desviación entre los bordes detectados y los ideales representados en lo que se conoce como "ground-truth" o bordes ideales. Se define como:

$$FOM = \frac{1}{max(I_I, I_A)} \sum_{i=1}^{I_A} \frac{1}{1 + \delta e^2(i)}, \tag{3.12}$$

donde $I_A$ es el número de píxeles detectados como bordes, $I_I$ es el número de píxeles definidos idealmente como bordes, $e(i)$ es la distancia entre los bordes obtenidos y los ideales y $\delta$ es un factor de escala que se fija normalmente como $\frac{1}{9}$.

Como se comentó previamente es sencilla de implementar y rápida en ejecución pero tiene el problema de la generación del "ground-truth" de las imágenes a valorar. Esto último no suele ser problemático en imágenes con formas geométricas definidas pero sí que lo es en imágenes reales donde la subjetividad es mayor a la hora de definir los bordes ideales.

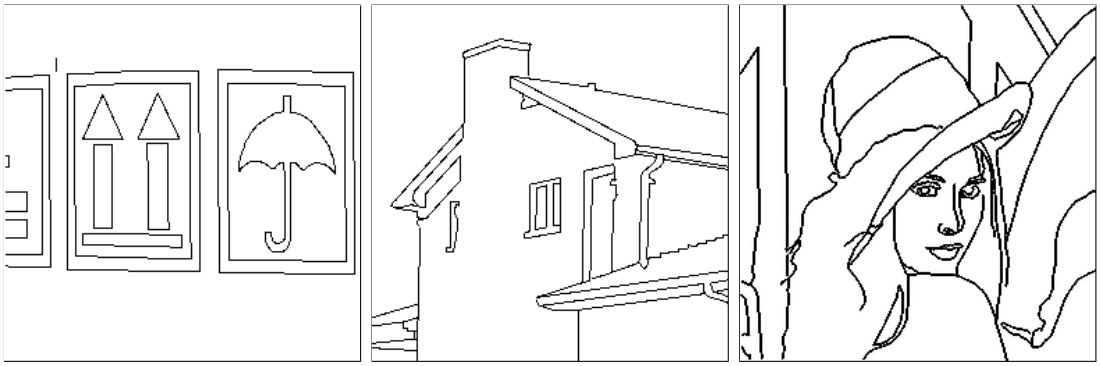
**Sobresegmentación** Con este índice se pretende complementar la información ofrecida por el FOM indicando si el método utilizado produce más píxeles de borde de los establecidos el ideal. La medida se define como:

$$SOB = \frac{\sum_n G \wedge I}{\sum_n G \vee I}, \tag{3.13}$$

donde $G$ es la imagen de "ground-truth" e $I$ es la imagen de bordes, siendo ambas imágenes lógicas donde los bordes se marcan con 1 y los no bordes con 0. Este índice representa, por tanto, la relación entre el número de píxeles en los que las dos imágenes coinciden y la suma de píxeles totales si se unen las dos. Un índice cercano a 1 indicará coincidencia entre ambas imágenes, mientras que uno cercano a 0 indicará no coincidencia o sobresegmentación.

### 3.3.2. Resultados.

En este apartado se presentarán resultados numéricos usando los dos índices definidos previamente. Con estos resultados se pretenden comparar, por un lado, los métodos pro-



(a) (b) (c)

Figura 3.44: Ejemplo de imágenes "ground-truth" usadas en las medidas de calidad de los detectores de bordes.

puestos y sus diferentes variantes y, por otro lado, estos con el método de Canny que es el que se toma como referencia en en diferentes trabajos de detección de bordes ([Heath97]). Para realizar esas medidas se han utilizado las imágenes Pattern, Oldhouse y Lenna sobre las que se han ido presentado resultados a lo largo del capítulo. Sobre ellas se definieron los "ground-truth" necesarios como puede verse en la figura 3.44.

Los métodos de marcado de píxeles que se compararán serán:

- Búsqueda de máximos en horizontal y vertical por separado (Max).
- Búsqueda de máximos en la imagen de bordes con una limpieza posterior con procesado morfológico (Max+Morf).
- Adelgazamiento de la imagen de bordes.
- Supresión de no máximos y umbralizado con histéresis (Nomax+Umbr).

Para el entrenamiento de las SVM se utilizaron imágenes de tamaño $16 \times 16$ con una diferencia de valores de gris de 50 y un kernel gaussiano con una $\gamma = 8 \cdot 10^{-5}$ y $C = 1000$.

**Resultados usando valores de gris.** En primer lugar se compararán los métodos usando exclusivamente valores de gris de la imagen en una ventana de $3 \times 3$ alrededor de cada píxel. Posteriormente se realizarán pruebas variando el tamaño de la ventana y usando también diferencias.

En la tabla 3.6 se presentan dichos resultados para los métodos de marcado definidos previamente y además se añaden datos sobre la diferencia entre usar normalización y escalado o función de probabilidad para obtener la imagen de gradiente de bordes. Se muestran también los valores obtenidos para el algoritmo de Canny.

Hay que destacar la diferencia existente entre los resultados de las 3 imágenes utilizadas. En el caso de la imagen Pattern los métodos basados en SVM obtienen mejores resultados que el algoritmo de Canny. En el caso de Oldhouse sólo se supera al algoritmo de Canny en el índice FOM pero no en el de sobresegmentación. Sin embargo, para la imagen Lenna es el algoritmo de Canny el que obtiene los mejores resultados. Parece que

Tabla 3.6: Resultados de segmentación con una ventana es de $3 \times 3$ y utilizando los valores de gris de la imagen. Los mejores resultados se marcan en negrita.

| | | Pattern | | Oldhouse | | Lenna | |
|---|---|---|---|---|---|---|---|
| Método \ Medida | | FOM | SOB | FOM | SOB | FOM | SOB |
| Canny | | 0.9776 | 0.4073 | 0.9077 | **0.3438** | **0.7893** | **0.1246** |
| Max | NormEsc | 0.9661 | **0.5655** | 0.8637 | 0.2660 | 0.6966 | 0.0882 |
| | Prob | 0.9724 | 0.5499 | 0.8940 | 0.2959 | 0.7434 | 0.0898 |
| Max+Morf | NormEsc | 0.9607 | 0.5418 | 0.8922 | 0.2760 | 0.7412 | 0.0919 |
| | Prob | **0.9824** | 0.5602 | 0.8942 | 0.2881 | 0.7439 | 0.0935 |
| Adelgazamiento | NormEsc | 0.9412 | 0.5192 | 0.9316 | 0.3099 | 0.7357 | 0.0918 |
| | Prob | 0.9412 | 0.5072 | **0.9427** | 0.3205 | 0.7341 | 0.0911 |
| Nomax+Umbr. | NormEsc | 0.8551 | 0.4523 | 0.8125 | 0.2425 | 0.6828 | 0.0811 |
| | Prob | 0.8561 | 0.4750 | 0.7951 | 0.2100 | 0.6706 | 0.0737 |

en el caso de imágenes con formas geométricas definidas los algoritmos propuestos pueden incluso sobrepasar al estándar de Canny.

De los métodos propuestos, los de búsqueda de máximos y el de adelgazamiento son los que obtienen mejor calificación. El peor, y posiblemente descartable para su uso posterior, es el de supresión de no máximos y umbralizado.

En cuanto al uso de normalización y escalado o función de probabilidad no hay un dato concluyente a favor de uno u otro. Dependiendo del método de marcado y de la imagen se deberían probar ambos métodos para obtener el mejor resultado. Sin embargo, hay que destacar que los dos mejores resultados de los métodos propuestos se dan usando la función de probabilidad.

**Resultados usando diferencias.** En este apartado se muestran datos de detección de bordes utilizando las diferencias con el píxel central en lugar de los valores de gris de la imagen. Como se vio en la sección dedicada al entrenamiento (3.2.1), el uso de diferencias permitía reducir el número de vectores soporte y con ello aumentar la velocidad; quedaba, sin embargo, determinar si la calidad se reducía en la misma medida o los resultados se mantenían. En la tabla 3.7 aparecen resultados numéricos y en los que se han marcado los mejores en negrita. Hay que decir que en este caso se muestran sólo valores referidos al uso de la función de probabilidad para simplificar la comparación y porque, en general, eran mejores. Comparando los resultados con los de la tabla 3.6 puede observarse una disminución en cuanto a calidad aunque no como para descartar el uso de diferencias cuando se requiera un aumento de velocidad.

**Resultados usando un tamaño de ventana de $5 \times 5$.** Por último, queda por determinar la calidad en el caso de aumentar la ventana alrededor del píxel en el que se quiere detectar un borde. En la tabla 3.8 aparecen los valores obtenidos usando una ventana de $5 \times 5$ en lugar de la de $3 \times 3$. Nuevamente se muestran sólo valores a partir de la función de probabilidad y además usando los valores de gris de la imagen.

Tabla 3.7: Resultados de segmentación con una ventana es de $3 \times 3$ y utilizando las diferencias con el píxel central. Imagen de gradiente obtenida con la función de probabilidad. Los mejores resultados aparecen en negrita.

| | Pattern | | Oldhouse | | Lenna | |
|---|---|---|---|---|---|---|
| Medida / Método | FOM | SOB | FOM | SOB | FOM | SOB |
| Canny | **0.9776** | 0.4073 | 0.9077 | **0.3438** | **0.7893** | **0.1246** |
| Max | 0.9608 | 0.2675 | 0.9134 | 0.2229 | 0.7371 | 0.0977 |
| Max+Morf | 0.9700 | 0.3210 | 0.9338 | 0.2487 | 0.7341 | 0.0866 |
| Adelgazamiento | 0.9523 | **0.4428** | **0.9406** | 0.2974 | 0.7380 | 0.1011 |
| Nomax+Umbr. | 0.8416 | 0.1642 | 0.7974 | 0.1476 | 0.7025 | 0.0800 |

Tabla 3.8: Resultados de segmentación con una ventana es de $5 \times 5$ y utilizando los valores de gris de la imagen. Se ha utilizado la función de probabilidad. Los mejores valores se marcan en negrita.

| | Pattern | | Oldhouse | | Lenna | |
|---|---|---|---|---|---|---|
| Medida / Método | FOM | SOB | FOM | SOB | FOM | SOB |
| Canny | **0.9776** | 0.4073 | 0.9077 | **0.3438** | **0.7893** | **0.1246** |
| Max | 0.7653 | 0.1533 | 0.7532 | 0.1518 | 0.7221 | 0.0821 |
| Max+Morf. | 0.9687 | 0.2738 | **0.9351** | 0.2087 | 0.7291 | 0.0791 |
| Adelgazamiento | 0.8925 | **0.4546** | 0.8452 | 0.2180 | 0.7131 | 0.0881 |
| Nomax+Umbr. | 0.8072 | 0.2291 | 0.7635 | 0.1580 | 0.6875 | 0.0821 |

Hay que destacar la reducción de calidad respecto a los valores de la tabla 3.6. Además, un dato importante es el aumento de número de vectores soporte como ya se demostró en la tabla 3.3. Esto último implica un mayor tiempo de ejecución aumentado además por el hecho de que los vectores tienen 25 componentes en vez de 9. Según estos resultados no parece útil el aumentar la ventana de detección puesto que empeora la calidad y aumenta el tiempo de ejecución.

## 3.4. Resumen

En este capítulo se ha presentado una nueva técnica de detección de bordes basada en el uso de máquinas de vectores soporte en la que la obtención de los píxeles candidatos a ser borde no se realiza mediante la obtención del gradiente sino mediante el entrenamiento con imágenes sintéticas y la posterior clasificación de los vectores de las imágenes. Es esta una primera característica diferenciadora, pues el entrenamiento sintético permite elegir los parámetros más adecuados para la detección ya en la fase inicial y no en el postprocesado. De esta manera se permite una mejor adecuación al tipo de imágenes sobre las que se está trabajando.

Siguiendo con el entrenamiento, se ha realizado un estudio sobre los kernel disponibles y los resultados que se obtienen, así como, la selección de características para permitir, si es necesario, reducir la dimensión de los vectores utilizados en la clasificación de los píxeles y, por tanto, la velocidad. Se ha llegado a la conclusión de que los kernel más adecuados serían el polinomial y el gaussiano, siendo sus principales características, la rapidez en el caso polinomial y la reducción de ruido en el caso gaussiano.

La información obtenida de la clasificación de los píxeles en "borde" o "no borde" puede ser útil en determinados casos, aunque el hecho de que los bordes, generalmente, no sean abruptos hace que los bordes detectados de esa manera no sean suficientemente definidos y deban ser postprocesados. Para ese postprocesado hay que obtener una gradación de los bordes y no ya su clasificación. Para ello se proponen dos métodos, uno en el que la salida de las SVM se traduce linealmente a una función cuya salida esta entre 0 y 255 y otro en el que esa traducción se realiza de manera estadística obteniendo una función de probabilidad. Ambos métodos se comparan y se constata el alto grado de sensibilidad del método de probabilidad. Sin embargo, sus resultados de calidad con el índice FOM son buenos y, a veces, superiores a los obtenidos con el método de Canny.

El postprocesado necesario tras obtener la imagen de gradación de bordes también se puede enfocar de distintas maneras. Por un lado se puede usar una imagen de gradación bordes en la que aparezcan marcadas todas las direcciones y por otro usar imágenes de bordes horizontales y verticales según aparece en varios métodos en el estado del arte. Para el primer caso se proponen una umbralización seguida de la aplicación de un algoritmo de adelgazamiento o bien umbralización y búsqueda de máximos. Esta última posibilidad debe ir acompañada después de un refinado pues aparecen bordes espúreos al estar todas las direcciones en la misma imagen. Para este refinado se define un procesado morfológico basado en SVM que produce buenos resultados de manera rápida. En el caso de utilizar por separado bordes horizontales y verticales se puede usar la técnica de supresión de no máximos e histéresis (ya usada por el algoritmo de Canny) o bien una búsqueda de

máximos en ambas direcciones y su posterior unificación; en este caso no aparecen bordes espúreos. En ambos casos se realiza un umbralizado inicial.

Por último se ha demostrado que la adición de ruido en el entrenamiento mejora la detección de los bordes, independientemente de la técnica utilizada dentro de las presentadas previamente. Esto hace que se pueda hablar de un mecanismo de resistencia al ruido en la detección de bordes con SVM. Este mecanismo está presente en muchos métodos del estado del arte pero normalmente empleando un filtrado inicial para suavizar la imagen y reducir el ruido.

# Capítulo 4

# Segmentación de imágenes en color

## 4.1. Estado del arte

En el apartado de detección de bordes se ha visto que una parte importante de algoritmos de visión parten de un primer paso en el que se identifican y *segmentan* los distintos objetos que componen la imagen para después poder tratarlos individualmente. En la detección de bordes se busca marcar las discontinuidades en la iluminación de la imagen para así detectar las fronteras de los distintos objetos. Sin embargo, usar sólo los niveles de gris de la iluminación es perder mucha de la información que nos aporta la imagen, concretamente no hacemos uso de los colores de cada uno de los elementos. Para la visión humana el color es una fuente importante de información y, muchas veces, nos permite distinguir los distintos objetos. También para la visión artificial es un dato importante y los algoritmos que se van a presentar a continuación pretenden hacer uso del color para conseguir una mejor separación de los distintos elementos de la imagen.

La literatura científica dedicada a este tema es muy amplia y difícilmente podemos presentarla en su totalidad en esta tesis. Para tener una idea clara de los diferentes tipos de algoritmos de segmentación basados en el color vamos a usar como base una serie de trabajos que precisamente hacen una revisión del estado del arte en este tema como son [Lucchese01, Cheng01]. En dichos trabajos se clasifican los algoritmos de segmentación (tanto de color como de niveles de gris) en tres grupos fundamentales:

- **Técnicas basadas en el espacio de características**. Puesto que suponemos que los objetos de interés van a estar agrupados según un determinado color y que además este color es la propiedad que los va ha caracterizar, el trabajo de segmentación se ha de desarrollar en el espacio de color que se elija convenientemente. Es decir, no trabajaremos directamente sobre los valores espaciales de la imagen sino en el espacio de características que nos interesa, que en este caso, es el del color. Su principal característica es que no usan la relación espacial entre los valores de color correspondientes. Dentro de estas técnicas encontramos las basadas en clustering, clustering adaptativo y umbralización del histograma.

- **Técnicas basadas en el dominio de la imagen**. En las técnicas basadas en el espacio de características podemos asegurar que las regiones segmentadas serán uniformes según un criterio basado en la característica utilizada (color). Sin embargo, no podemos asegurar que exista una homogeneidad espacial de dichas regiones, lo

cual es deseable desde el punto de vista de la segmentación, puesto que los puntos de la imagen pertenecientes a un mismo objeto están cercanos espacialmente. Si intentamos evitar este problema centrándonos sólo en la homogeneidad espacial puede ocurrir que marquemos como pertenecientes a un objeto puntos que no lo son pero que sí están próximos a él. Los métodos basados en estas técnicas intentan mezclar lo mejor de los dos criterios tratando de buscar regiones homogéneas tanto espacialmente como según un determinado criterio, en nuestro caso el color. Dentro de este grupo podemos encontrar técnicas conocidas como de Separar y mezclar (Split&Merge), de crecimiento de regiones, basados en detección de bordes y métodos basados en redes neuronales.

- **Métodos basados en propiedades físicas**. Estos últimos métodos están basados en el uso de modelos físicos para la interacción de la luz con las superficies coloreadas. Estos métodos intentan evitar el problema creado por los cambios de luz a la hora de buscar regiones homogéneas en la imagen. En general, estos métodos plantean modelos de reflexión para diferentes materiales y con ciertos puntos de iluminación. De esta manera se puede encontrar una escena en la que se eviten los problemas derivados de las distintas reflexiones según los distintos ángulos de incidencia y las distintas fuentes de luz.

Cada uno de los grupos presentados tiene unas ventajas y unos inconvenientes que les hacen adecuados para unas determinadas tareas. Los métodos basados en las propiedades físicas tienen como principal ventaja el hecho de que los cambios de iluminación no afectan a la clasificación de las zonas de la imagen. Sin embargo, sólo se han aplicado en entornos controlados (nunca en imágenes reales) en los que se conocen las fuentes de luz y los materiales de los que se componen los objetos de la imagen. Además necesitan de muchos cálculos para su aplicación. Los métodos basados en el espacio de características tienen a su favor la rapidez de ejecución y el hecho de que se puedan aplicar todo tipo de técnicas de clustering y clasificación ampliamente conocidas. Su principal problema es la no conexión espacial de las zonas en las que se divide la imagen de manera que se tiende a la sobresegmentación debido a las diferentes condiciones de iluminación. Los métodos basados en el dominio de la imagen reducen ese problema intentando agrupar pixels en función de relaciones espaciales y de características. Su principal desventaja es un mayor tiempo de ejecución a costa de reducir el problema de la sobresegmentación y mejorar la relación espacial de las zonas de la imagen.

### 4.1.1. Espacios de color

Puesto que la característica fundamental que se va a utilizar para la segmentación es el color, es necesario definir cual es el espacio de color sobre el se va a trabajar puesto que esa elección será esencial a la hora del correcto funcionamiento del método elegido. El color es percibido por el ser humano como una combinación de tres estímulos $R$ por el rojo, $G$ por el verde y $B$ por el azul. Mediante la mezcla aditiva de estos colores se pueden conseguir la gran mayoría de los posibles. Esta mezcla aditiva es la que usan los monitores de ordenador o las televisiones para mostrar los distintos colores. Partiendo de las componentes $R, G, B$ se pueden derivar otras formas de representación del color (espacios)

usando transformaciones lineales o no lineales. Por tanto, el espacio $RGB$ será el punto de partida para la obtención de otros muchos espacios con sus ventajas e inconvenientes.

Muchos autores han intentado determinar cual es el espacio de color más apropiado para su método específico de segmentación [Cheng01, Gauch92]. Sin embargo, no existe un espacio de color que dé resultados satisfactorios en todos los tipos de imagen. Por ello, hay autores que proponen la utilización de un espacio de color híbrido usando componentes de distintos espacios y adaptando las componentes a la imagen que está siendo analizada [Vandenbroucke98, Vandenbroucke00, Busin04]. Aunque no existe una solución adaptable a todas las circunstancias.

Los espacios de color se pueden clasificar según varios criterios, a continuación se muestra la clasificación propuesta por [Vandenbroucke98] en cuatro grandes familias:

- **Espacios primarios**. Son aquellos que se basan en la teoría tricromática, asumiendo que es posible crear cualquier color mezclando cantidades apropiadas de tres colores primarios. Algunos de estos espacios son el $(RGB)$, $(rgb)$ y $(XYZ)$.

- **Espacios de luminancia-crominancia**. En estos espacios se trata de separar la información de color de la de intensidad para que el color no se vea afectado por los cambios de iluminación. Aparecerá entonces una componente de luminancia y otras dos de crominancia. Algunos espacios que realizan esta operación son $(YIQ)$, $(YUV)$, $(L^*a^*b^*)$ y $(L^*u^*v^*)$.

- **Espacios de percepción**. Estos espacios separan la información de color tratando de cuantificar la forma de percepción del color mediante la obtención de la intensidad, el tono y la saturación. Los espacios de este grupo son el $(HSI)$ y sus derivados $(HSB)$, $(HSL)$, y $(HSV)$ [Tsang96, Tepichin-Rodriguez95, Kim96a].

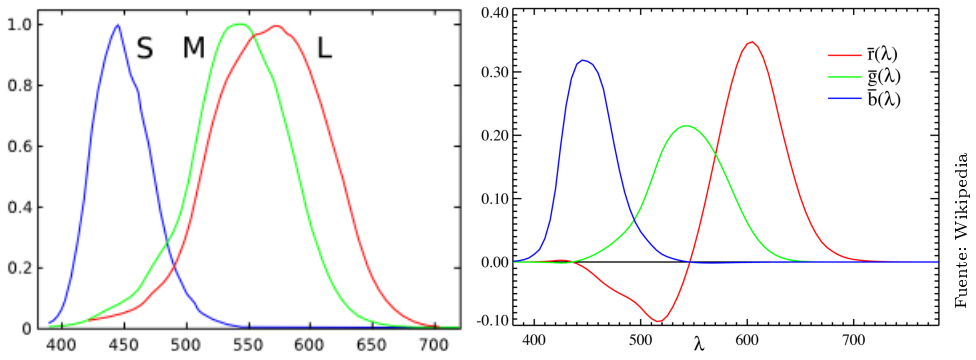
- **Espacios de componentes estadísticamente independientes**. Son el resultado de aplicar distintos métodos estadísticos para obtener unas componentes lo más incorreladas posible. Uno de estos espacios es el $(I_1 I_2' I_3')$ definido por Ohta en [Ohta80].

A continuación se presentan algunos de estos espacios indicando como obtenerlos y sus principales ventajas e inconvenientes.

**Espacio $RGB$.** Como se ha comentado anteriormente, el ser humano percibe los colores como combinación aditiva de tres componentes $R, G, B$ [Plataniotis00]. Las componentes $R, G$ y $B$ se pueden representar como los valores de brillo de una imagen después de haber sido filtrada por tres filtros distintos (filtros de rojo, verde y azul) basados en las siguientes ecuaciones:

$$\begin{aligned} R &= \int_\lambda E(\lambda)\,\bar{r}(\lambda)\,d\lambda, \\ G &= \int_\lambda E(\lambda)\,\bar{g}(\lambda)\,d\lambda, \\ B &= \int_\lambda E(\lambda)\,\bar{b}(\lambda)\,d\lambda, \end{aligned}$$

donde $\bar{r}, \bar{g}$ y $\bar{b}$ son los filtros de color para una luz incidente de valor $E(\lambda)$ y $\lambda$ es la longitud de onda. Los filtros se han diseñado para simular de alguna manera el comportamiento de los conos de la retina que se agrupan en tres tipos dependiendo del color predominante al que reaccionan (Figura 4.1).




**(a)** **(b)**

Figura 4.1: Comparación de la respuesta al color de los conos del ojo (a) y los filtros del espacio RGB (b)

El espacio *RGB* puede ser representado geométricamente en un cubo tridimensional (Figura 4.2). Las coordenadas de cada punto dentro del cubo representan los valores de las componentes rojo, verde y azul de cada color. El espacio *RGB* es el usado en las televisiones (para reproducir las imágenes) y en las cámaras digitales. Los monitores de vídeo reproducen las imágenes en color dando más o menos intensidad a cada uno de los colores primarios que contiene cada pixel de la imagen.

Este espacio es adecuado para reproducir el color pero tiene limitaciones para el análisis y la segmentación de imágenes debido a la alta correlación existente entre sus componentes [Littmann97]. La alta correlación implica que un cambio en la intensidad modifica las tres componentes, no siendo, por tanto, independientes entre sí. Por otro lado las diferencias de color medidas en el espacio *RGB* no son uniformes y, por tanto, no es posible medir la similitud entre dos colores mediante su distancia en el espacio *RGB*.

**Espacio rgb.** Cuando se busca el espacio de color más adecuado para la segmentación, es conveniente, que los cambios de iluminación no afecten a la información de color. De esta manera se podrá segmentar la misma escena, con los mismos resultados, independientemente de la iluminación. Esto no ocurre, por ejemplo, con el espacio *RGB*. Un método eficiente es conseguir que las variaciones de la intensidad sean uniformes a lo largo de toda la distribución espectral de color. Para ello se realiza una transformación no lineal sobre el espacio *RGB* que permite obtener un espacio de color normalizado, según las siguientes expresiones:

$$r = R/\left(R+G+B\right),$$
$$g = G/\left(R+G+B\right),$$
$$b = B/\left(R+G+B\right).$$

Observando dichas expresiones se ve claramente que $r+g+b=1$, y que teniendo dos de las componentes la tercera de ellas puede ser obtenida directamente. Podemos entonces utilizar sólo dos de esas componentes para la segmentación.

La normalización hace que este espacio sea relativamente robusto a cambios de iluminación. Pero un problema evidente es que las componentes son sensibles al ruido para

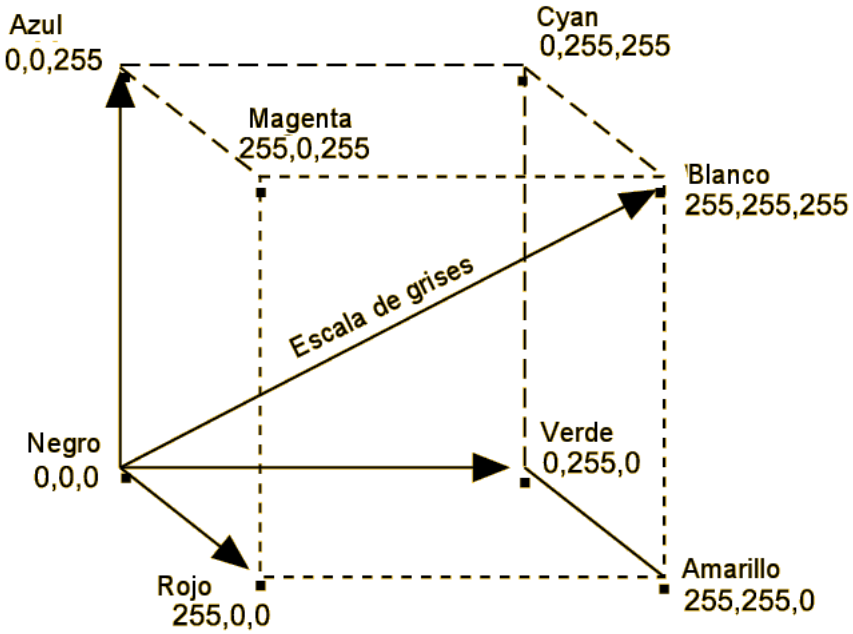



Figura 4.2: Espacio de color RGB representado en un cubo tridimensional

valores bajos de intensidad, puesto que estamos dividiendo por cantidades pequeñas en la normalización.

**Espacio XYZ.** El CIE (Commission International de l'Eclariage) ha diseñado varios espacios de color que presenten uniformidad en la percepción humana y que, por tanto, tengan en cuenta las necesidades psicofisiológicas de los observadores [Plataniotis00]. Todos ellos tienen como punto de partida tres componentes primarias llamadas $X, Y$ y $Z$. Sus ecuaciones serían:

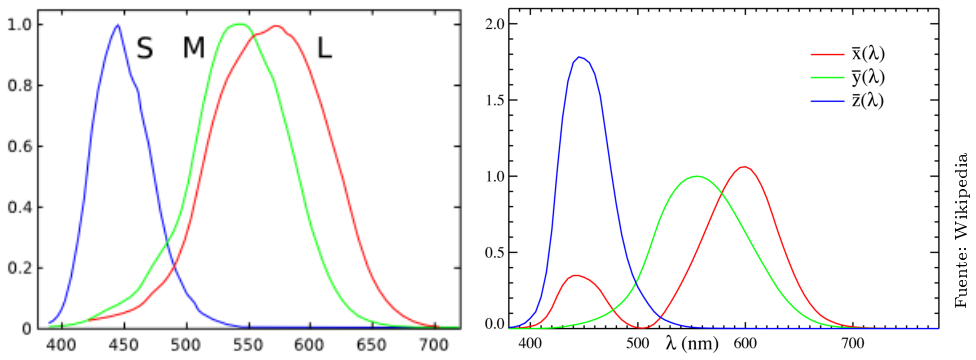
$$X = \int_\lambda E(\lambda)\,\bar{x}(\lambda)\,d\lambda,$$
$$Y = \int_\lambda E(\lambda)\,\bar{y}(\lambda)\,d\lambda,$$
$$Z = \int_\lambda E(\lambda)\,\bar{z}(\lambda)\,d\lambda,$$

donde $\bar{x}, \bar{y}$ y $\bar{z}$ son los filtros de color para una luz incidente de valor $E(\lambda)$ y $\lambda$ es la longitud de onda. La aproximación al comportamiento fisiológico de la retina es mejor que en el caso $RGB$ (Figura 4.3). Cualquier color puede ser especificado como combinación de esas tres componentes.

Los valores de $X, Y$ y $Z$ se obtienen mediante una transformación lineal del espacio $RGB$. La matriz de transformación fijada por el CIE para un blanco de referencia E es la siguiente:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{4.1}$$

Esta transformación no es única y dependiendo del blanco de referencia usado puede cambiar, y de hecho, es distinta para los sistemas de televisión NTSC o PAL.




(a) (b)

Figura 4.3: Comparación de la respuesta al color de los conos del ojo (a) y los filtros del espacio XYZ (b)

El espacio $XYZ$ así conseguido sigue teniendo los problemas del $RGB$, variación con la iluminación y no uniformidad en la medida de distancias por lo que no se adecúa del todo a la tarea de la segmentación.

**Familia de espacios HSI.** Los espacios $HSI$, $HSV$, $HSL$ y similares pertenecen a una familia de espacios caracterizados porque intentan desacoplar la información referente al color de la referente a la iluminación, aunque en cada modelo lo hacen de una manera distinta. Son muy utilizados en procesado de imagen porque, en teoría, sí que consiguen separar la información de color y, por lo tanto, sólo se debería tener que utilizar una componente y no tres como en otros espacios de color. Otra ventaja de estos espacios es que tienen más relación con la forma en que la vista percibe las imágenes, por una parte el color, y por otra la intensidad luminosa [Carron94, Kim96b, Plataniotis00].

La información de color obtenida en el espacio $HSI$ se representa mediante las componentes de tono (Hue) y saturación (Saturation) mientras que la información de intensidad se obtiene de la cantidad de luz de la imagen. El tono representa el color puro y se corresponde con la longitud de onda predominante en la distribución espectral de la luz proveniente de la imagen. La saturación mide la pureza del color y representa la mezcla de luz blanca con el color dominante.

El espacio $HSI$ tiene coordenadas cilíndricas en las que el tono ($H$) es considerado como el ángulo formado por el color en el espacio $RGB$ y una línea de referencia. Normalmente se considera su representación como un doble cono en coordenadas cilíndricas (Figura 4.4). El rango de valores de $H$ va desde $0^o$ a $360^o$. Por ejemplo el color azul es $240^o$, el amarillo $60^o$, el verde $120^o$ y el magenta $300^o$. La saturación es la distancia radial desde el eje del doble cono. Cuanto más cercano al eje más mezcla de luz blanca tiene el color. La intensidad es la altura en la dirección del eje central. El eje contiene los niveles de gris desde intensidad 0 (negro) hasta intensidad 1 (blanco). En este espacio, la información de color, tal y como se percibe por los humanos, está representada únicamente por la componente de tono (H) puesto que variaciones de las otras dos componentes para un tono fijo no modifican sustancialmente la percepción del color (salvo en los valores extremos).

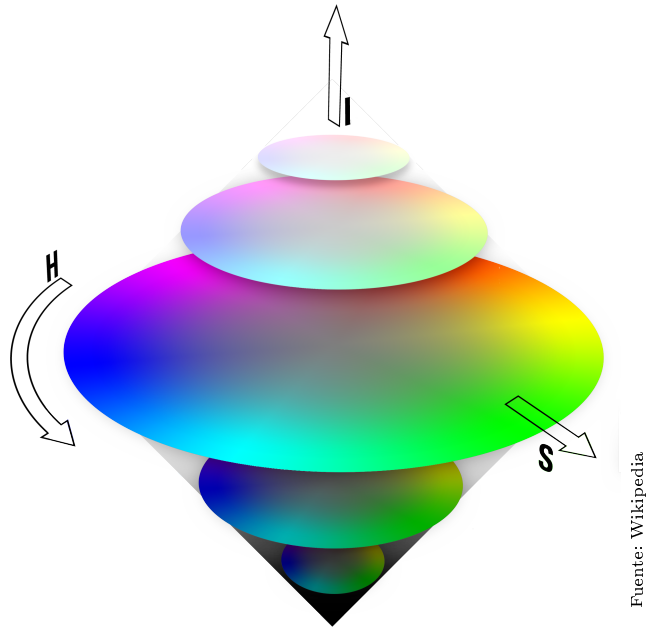


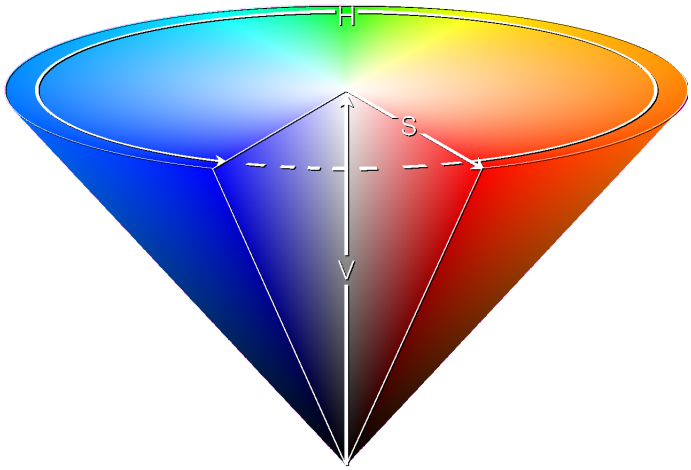

Figura 4.4: Espacio de color HSI representado en un doble cono tridimensional.

Las coordenadas $HSI$ se obtienen a partir del espacio $RGB$ con las siguientes fórmulas:

$$
\begin{aligned}
H_1 &= \arccos\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right), \\
H &= H_1 && \text{, si } B \leq G, \\
H &= 360^\circ - H_1 && \text{, si } B > G, \\
I &= \frac{(R+G+B)}{3}, \\
S &= 1 - \frac{\min(R,G,B)}{I}.
\end{aligned}
$$

Puede comprobarse que la componente de tono (H) está indefinida si la saturación es nula y la saturación está indefinida si la intensidad es nula. Este es uno de los principales problemas del uso de este tipo de espacios de color.

Otras variaciones de este espacio (como el $HSV$) se caracterizan porque mientras que el cálculo de la componente de tono se realiza de la misma forma, las otras dos componentes se obtienen de forma diferente. En el caso del espacio $HSV$ las ecuaciones que permiten




Fuente: Wikipedia

Figura 4.5: Espacio de color HSV representado en un cono tridimensional

obtener las componentes son:

$$
\begin{aligned}
H_1 &= \arccos\left(\frac{\frac{1}{2}(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right),\\
H &= H_1 \qquad\qquad\qquad\qquad ,\text{si } B \leq G,\\
H &= 360^{\text{o}} - H_1 \qquad\qquad ,\text{si } B > G,\\
S &= \frac{\text{máx}(R,G,B) - \text{mín}(R,G,B)}{\text{máx}(R,G,B)},\\
V &= \frac{\text{máx}(R,G,B)}{255}.
\end{aligned}
$$

Este espacio se basa en la terminología y el empleo que hacen de los colores los artistas y fue propuesto originalmente en [Smith78] y puede ser representado por un cono en coordenadas cilíndricas (Figura 4.5).

Las ventajas que caracterizan a los espacios de la familia $HSI$ son:

- Buena compatibilidad con la percepción humana.
- Posibilidad de separación de valores cromáticos de valores acromáticos.
- La opción de usar una característica del color, tono, solamente para la segmentación. Muchos métodos de segmentación se aprovechan de esto, ya que, si la segmentación se realiza en una componente del color en vez de tres los algoritmos pueden ser mucho más rápidos.
- Además, el tono es invariante a los cambios de iluminación, y por tanto estos cambios en una imagen no afectarían a la segmentación obtenida.

Sin embargo, los espacios de color orientados al tono tienen algunas desventajas significativas, por ejemplo:

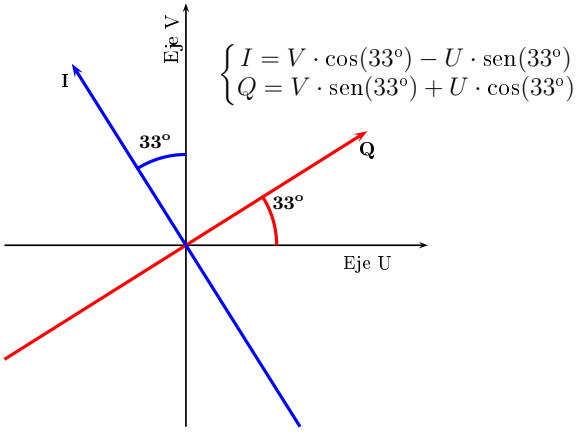



Figura 4.6: Ejes de coordenadas IQ y su equivalencia con UV

- Singularidades en la transformación, por ejemplo, tonalidad indefinida para los puntos acromáticos.
- Gran sensibilidad a las pequeñas variaciones de $RGB$ cerca de los puntos singulares. Produciéndose grandes saltos para variaciones pequeñas de las componentes $RGB$.
- Inestabilidad numérica al trabajar con el tono debido a la naturaleza angular del mismo.
- Además, si la intensidad cae cerca del blanco o del negro, el tono y la saturación no cuentan apenas para distinguir los colores.

**Espacios YIQ e YUV.** Estos espacios de color están especialmente diseñados para la transmisión de la información de color en los sistemas de televisión. El espacio $YIQ$ se utiliza en el sistema de televisión NTSC y el $YUV$ para el sistema PAL. Ambos están diseñados para minimizar el ancho de banda de transmisión, puesto que en dicha transmisión se tiene en cuenta que el sistema visual humano es más sensible a los cambios de iluminación ($Y$) que a los de color, por tanto, la información de color puede ser transmitida con un ancho de banda más reducido.

Otro aspecto que se tuvo en cuenta es el experimento de König (1894). De él se obtiene que el sistema visual humano no tiene la misma sensibilidad para todos los colores. Es decir, que existen unas tonalidades que el ojo distingue con mayor facilidad y otras que apenas ve. Los colores mejor apreciados por el ojo son el naranja y el cian. Los dos se encuentran sobre el mismo eje (pero en sentidos opuestos), formando un ángulo de 33º con el eje V, correspondiente a la coordenada V del modelo YUV. A partir de estos resultados se llegó a la conclusión de que era conveniente transmitir estos dos colores con mayor ancho de banda, ya que el ojo es capaz de apreciar zonas pequeñas de ellos, y mandar con un ancho de banda menor aquellos en los que el ojo sólo distingue zonas grandes. Esta fue la idea que dio origen a las nuevas coordenadas I (In phase) y Q (Quadrature) para el sistema NTSC. Así, se tomó el eje I en la dirección de los colores naranja y cian, y al eje Q perpendicular a éste, es decir, en la dirección de los colores magenta y verde (Figura 4.6) Los componentes de ambos espacios se pueden obtener mediante las siguientes

expresiones:

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.253 & -0.312 \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix}, \tag{4.2}$$

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.47 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix}, \tag{4.3}$$

donde las componentes $R', G', B'$ están normalizadas entre 0 y 1. Tanto las componentes $IQ$ como las $UV$ describen conjuntamente el tono y la saturación de la imagen. Al obtenerse mediante una transformación lineal, su cálculo es sencillo y rápido. Desde el punto de vista de la segmentación tienen la ventaja de que son necesarias sólo dos componentes en lugar de tres. Sin embargo, ambos espacios son no uniformes desde el punto de vista de la percepción y por tanto las distancias medidas en estos espacios tampoco son uniformes.

**Espacios L*a*b* y L*u*v*.** Los espacios de color presentados hasta ahora son todos no uniformes desde el punto de vista de la percepción. Sin embargo, la idea en la que se basan los espacios $L^*a^*b^*$ y $L^*u^*v^*$ es encontrar una transformación no lineal en la que, partiendo de los componentes $XYZ$, se obtenga un espacio en el que la distancia entre dos colores distintos sea proporcional a la distancia que aprecie un observador humano. De esta manera tendremos un espacio uniforme desde el punto de vista de la percepción. El CIE trabajó cerca de diez años para encontrar una transformación que cumpliera más o menos satisfactoriamente con las premisas anteriores. Finalmente, en 1976 estandarizó dos espacios, $L^*a^*b^*$ y $L^*u^*v^*$ como uniformes perceptualmente. Son ligeramente diferentes pero ambos son igualmente buenos en cuanto a la uniformidad conseguida.

Ambos espacios tienen una componente de iluminación $L^*$ que correspondería a la cantidad de luz percibida. Las otras dos componentes corresponderían a la información de color. Las ecuaciones para obtener el espacio $L^*a^*b^*$ se muestran a continuación:

$$\begin{aligned} L^* &= 116 \left( \sqrt[3]{\frac{Y}{Y_o}} \right) - 16, \\ a^* &= 500 \left[ \sqrt[3]{\frac{X}{X_o}} - \sqrt[3]{\frac{Y}{Y_o}} \right], \\ b^* &= 200 \left[ \sqrt[3]{\frac{Y}{Y_o}} - \sqrt[3]{\frac{Z}{Z_o}} \right], \end{aligned} \tag{4.4}$$

donde $\frac{X}{X_o} > 0.01, \frac{Y}{Y_o} > 0.01$ y $\frac{Z}{Z_o} > 0.01$ y $X_o, Y_o, Z_o$ son los valores del blanco estándar tomado como referencia.

El espacio $L^*u^*v^*$ se obtiene:

$$\begin{aligned} L^* &= 116 \left( \sqrt[3]{\frac{Y}{Y_o}} \right) - 16, \\ u^* &= 13L^* \left( u' - u_o \right), \\ v^* &= 13L^* \left( v' - v_o \right), \end{aligned} \tag{4.5}$$

donde $\frac{Y}{Y_o} > 0.01$ y $Y_o, u_o$ y $v_o$ son los valores del blanco estándar tomado como referencia. Además:

$$\begin{aligned} u' &= \frac{4X}{X+15Y+3Z}, \\ v' &= \frac{6Y}{X+15Y+3Z}. \end{aligned} \tag{4.6}$$

Como se puede comprobar ambas transformaciones son no lineales y dependen del blanco que se tome como referencia. Si se toma como blanco el punto $RGB = [1,1,1]$ (Normalizado) y lo transformamos a valores $XYZ$ obtenemos el siguiente blanco de referencia:

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} = \begin{bmatrix} 0.4125 & 0.3576 & 0.1804 \\ 0.2127 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9502 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{4.7}$$

Sin embargo, si nos basamos en el estándar de la European Broadcasting Union (EBU) la transformación sería la siguiente:

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} = \begin{bmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{4.8}$$

Los blancos estándar son, entonces, tantos como normas haya para la transformación del espacio $RGB$ al $XYZ$.

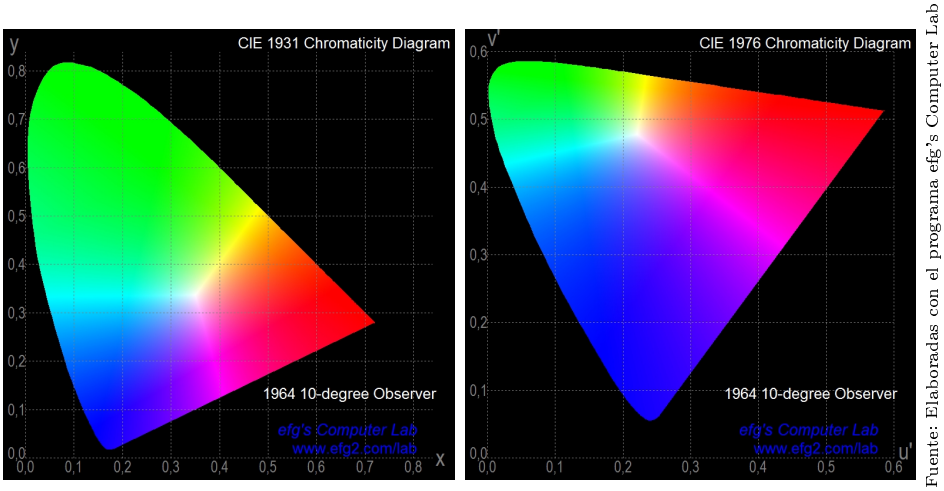
En la figura 4.7 se pueden visualizar los cambios entre el espacio $XYZ$ y el $L^*u^*v^*$ a través de sus diagramas de cromaticidad, que no son más que la proyección de ambos espacios en sólo dos coordenadas. En el espacio $XYZ$ aparece exagerado el espacio dedicado a los tonos verdes mientras que en $L^*u^*v^*$ no es así. Hay que señalar que estas transformaciones están basadas en la apreciación subjetiva de varios observadores a los que se les pidió medir la distancia que apreciaban entre varios colores, por tanto, la distribución que aparece en la figura 4.7(b) es la que realmente aprecia el observador humano.

Las ventajas del uso de estos espacios en la segmentación en color son:

- Separación de la información de intensidad y de color a diferencia de espacios como el $RGB$.
- La comparación entre dos colores se puede hacer mediante medida geométrica de distancias siendo ya medidas objetivas y no subjetivas.

En cuanto a las desventajas:

- Su cálculo es costoso y supone un importante tiempo de cálculo.
- Son transformaciones en las que pueden aparecer singularidades para valores bajos de intensidad.
- Para valores bajos de intensidad no son válidas las transformaciones presentadas y habría que calcularlas con una fórmula lineal




**(a)** **(b)**

Figura 4.7: Diagrama de cromaticidad para el espacio CIE $XYZ$ (a) y el CIE $L^*u^*v^*$ (b)

**Espacio $\mathbf{I_1I_2'I_3'}$ (Ohta).** En [Ohta80] se plantea la búsqueda de un espacio de color en el que las componentes estén incorreladas entre sí, de manera que se puedan usar por separado para la tarea de la segmentación. Para obtener dichas componentes se realiza la transformación de Karhunen-Loeve sobre un set de ocho imágenes en color elegidas por los autores. El proceso comienza por el cálculo de la matriz de covarianza de las distribuciones correspondientes a las componentes $RGB$ y sus correspondientes autovalores y autovectores. Los autovectores se ordenan según su autovalor asociado de manera que se obtienen tres nuevas componentes a partir de las $RGB$ en función de los autovectores asociados:

$$X_i = \omega_{Ri} \cdot R + \omega_{Gi} \cdot G + \omega_{Bi} \cdot B, \tag{4.9}$$

donde $\omega_{Ri}, \omega_{Gi}$ y $\omega_{Bi}$ son las componentes de los autovectores y además estos vectores tienen norma unidad.

Para cada imagen se obtienen distintos autovectores (Tabla 4.8) que son bastante dispares pero los autores los aproximan a unos valores que representan la tendencia general de la tabla, según la siguiente ecuación:

$$\begin{aligned} \mathbf{W}_1 &= \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)^t, \\ \mathbf{W}_2 &= \left(\pm\frac{1}{2}, 0, \mp\frac{1}{2}\right)^t, \\ \mathbf{W}_3 &= \left(-\frac{1}{4}, \frac{1}{2}, -\frac{1}{4}\right)^t. \end{aligned} \tag{4.10}$$

Por tanto, se pueden obtener unas nuevas componentes de color ortogonales a partir de la siguiente transformación:

Eigenvectors of $\Sigma$ for a Whole Image.[a]

| | $w_{R1}$ | $w_{G1}$ | $w_{B1}$ | $w_{R2}$ | $w_{G2}$ | $w_{B2}$ | $w_{R3}$ | $w_{G3}$ | $w_{B3}$ |
|---|---|---|---|---|---|---|---|---|---|
| Cylinder | 0.269 | 0.363 | 0.367 | 0.469 | 0.095 | − 0.437 | − 0.308 | 0.461 | − 0.231 |
| Building | 0.269 | 0.340 | 0.391 | 0.479 | 0.103 | − 0.418 | − 0.296 | 0.485 | − 0.219 |
| Seaside | 0.258 | 0.380 | 0.362 | − 0.585 | 0.056 | 0.358 | − 0.176 | 0.464 | − 0.360 |
| Girl | 0.336 | 0.354 | 0.309 | − 0.493 | 0.193 | 0.314 | − 0.094 | 0.474 | − 0.436 |
| Room | 0.193 | 0.341 | 0.467 | 0.612 | 0.079 | − 0.310 | − 0.209 | 0.507 | − 0.284 |
| Home | 0.197 | 0.328 | 0.476 | 0.492 | 0.180 | − 0.328 | − 0.313 | 0.484 | − 0.204 |
| Auto | 0.304 | 0.317 | 0.378 | 0.239 | 0.309 | − 0.452 | − 0.514 | 0.450 | 0.036 |
| Face | 0.175 | 0.411 | 0.414 | 0.523 | 0.128 | − 0.349 | − 0.295 | 0.416 | − 0.289 |

[a] Normalized as $w_G \geq 0, |w_R| + |w_G| + |w_B| = 1$.

Fuente: Tomada de [Ohta80]

Figura 4.8: Tabla de autovectores en los que se basan los cálculos del espacio $I_1 I_2' I_3'$

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \pm\frac{1}{2} & 0 & \mp\frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{4.11}$$

Observando dichas componentes se puede ver que $I_1$ es una componente de iluminación similar a la $I$ del espacio $HSI$. Por tanto, las otras dos componentes son las que llevan la información de color. En [Ohta80] se realizan distintas comparaciones de segmentación con otros espacios y sobre la importancia de cada uno de los nuevos componentes, llegando a la conclusión de que la componente más importante para la detección de bordes es la $I_1$ aunque si es necesario tener en cuenta cambios de color se necesitan las otras dos componentes. En todas esas pruebas los autores no usan el espacio de la ecuación 4.11 sino una modificación en que las componentes $I_2$ e $I_3$ son multiplicadas por 2 quedando:

$$\begin{bmatrix} I_1 \\ I_2' \\ I_3' \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 0 & -1 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{4.12}$$

Este espacio tiene como principales ventajas la facilidad de cálculo y la no correlación entre sus componentes aunque como el resto de espacios provenientes de una transformación lineal no es homogéneo en cuanto que las distancias geométricas no reflejan distancias de percepción.

### 4.1.2. Métodos de segmentación

Unas vez estudiados los diferentes espacios de color que pueden ser usados en la segmentación en color, se van a presentar algunos de los sistemas de segmentación usados en la literatura al respecto para observar ventajas, inconvenientes y características de estos. Después esta información será útil para la comparación con los métodos propuestos en esta tesis.

**Umbralización.** El sistema de umbralización es el más sencillo y rápido de implementar en el práctica puesto que sólo necesita de la obtención de una serie de umbrales que definan una zona del espacio de color en el que se encuentra el tipo de color que queremos detectar. La forma de obtener esos umbrales puede ser variada, yendo desde el método de "prueba y error" hasta la obtención automática de los mismos [Otsu79]. Al basarse sólo en el color y no en la proximidad espacial podemos clasificar este método en los que utilizan un "espacio de características".

Una técnica habitual es la extracción de las zonas de color que se quieren detectar de una imagen modelo y la visualización de los píxeles que los forman en el espacio de color que se esté utilizando. De esta forma se pueden ver las zonas del espacio que ocupan y obtener de manera empírica los valores de los umbrales. En la figura 4.9 se puede ver un ejemplo de esta técnica. En 4.9(a) aparece la imagen en la que se va a detectar el color con las zonas extraídas marcadas. En la imagen 4.9(b) aparecen los píxeles extraídos representados en el espacio de color RGB. A partir de esta última imagen se pueden extraer los umbrales para acotar el color dentro del espacio RGB, en el ejemplo mostrado esos umbrales serían:

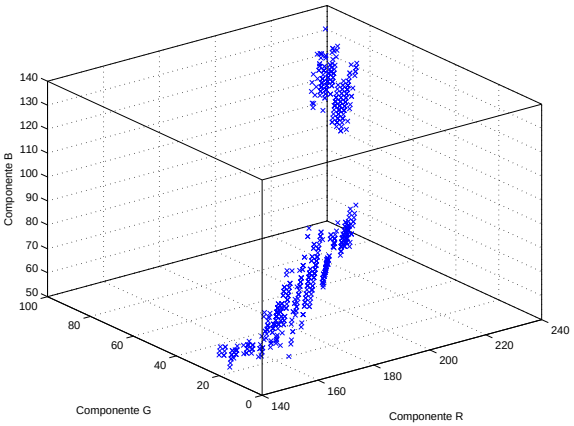
1. La componente R debe estar entre 140 y 240.
2. La componente G estará entre 0 y 100.
3. La componente B estará comprendida entre 50 y 140.

Usando dichos umbrales se marca la zona de color elegida según se muestra en la figura 4.9(c).

Como se ha comentado en el punto 4.1.1, la elección del espacio de color es fundamental a la hora de la segmentación, y el caso de la técnica de umbralización no es una excepción. En el ejemplo anterior, el objetivo era la segmentación de la zona de color rojo. Sin embargo, se puede observar que los bordes superiores e inferiores de esta zona no aparecen segmentados debido a que tienen unos valores de luminancia mayores o menores. Este es uno de los problemas del uso del espacio RGB, tenemos tres componentes para definir un color pero estas dependen de los cambios de iluminación y además están correladas entre sí. Una forma de evitar estos problemas es el uso del espacio HSV donde la información de color está concentrada en la componente H (Tono) y, en menor medida, en la S (Saturación). Un ejemplo de umbralizado usando el espacio HSV se presenta en la figura 4.10. Si se observa la representación en el espacio HSV (Figura 4.10(a)) se aprecia que todos los puntos están muy concentrados en la componente H mientras que en las otras dos hay más diferencia. Esto nos indica que esta componente es la que mejor va a separar los colores en este espacio. La figura 4.10(b) muestra como usando sólo la componente H se mejora la segmentación respecto al espacio RGB. Sin embargo, se aprecia también uno de los problemas de este espacio, que es la inestabilidad en zonas de baja saturación cómo las que presenta la bola negra. Para evitar estos problemas se puede incluir la componente S en la segmentación, teniendo cuidado de que su valor esté por encima de un mínimo ya que si no el valor de H es inestable y por tanto no significativo. Este sería el caso de la figura 4.10(c) donde se usan ambas componentes y desaparecen las zonas de la bola negra segmentadas erróneamente.

El principal problema que presenta el método de umbralización es que las zonas de los espacios de color que se obtienen no son todo lo exactas que se desearía puesto que su

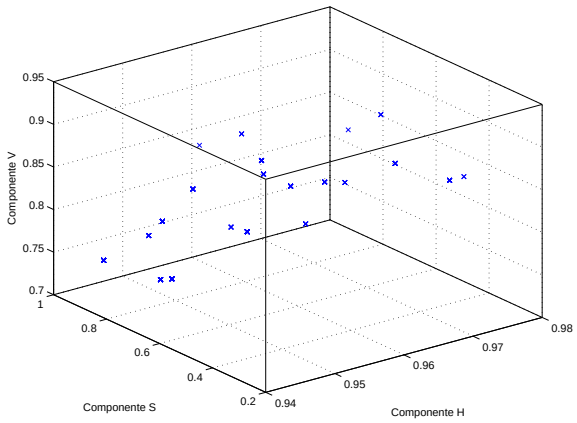
**(a)** ***Zonas extraídas.***




**(b)** ***Representación en el espacio RGB.***

**(c)** ***Resultado de la umbralización.***

Figura 4.9: Ejemplo de umbralizado en el espacio RGB.




(a) ***Representación en el espacio HSV de las zonas extraidas.***

(b) ***Resultado de la umbralización sólo con la componente H.***

(c) ***Resultado de la umbralización usando las componentes H y S.***

Figura 4.10: Ejemplo de umbralizado en el espacio HSV.

forma no es arbitraria sino definida por el tipo de umbral. De esta manera se pueden incluir zonas no deseadas en la segmentación. Además, tenemos el problema de la generalización pues lo que es válido para una imagen puede no serlo para otra. Estos problemas se pueden reducir con la conveniente elección del espacio de color pero no se evitarán por completo.

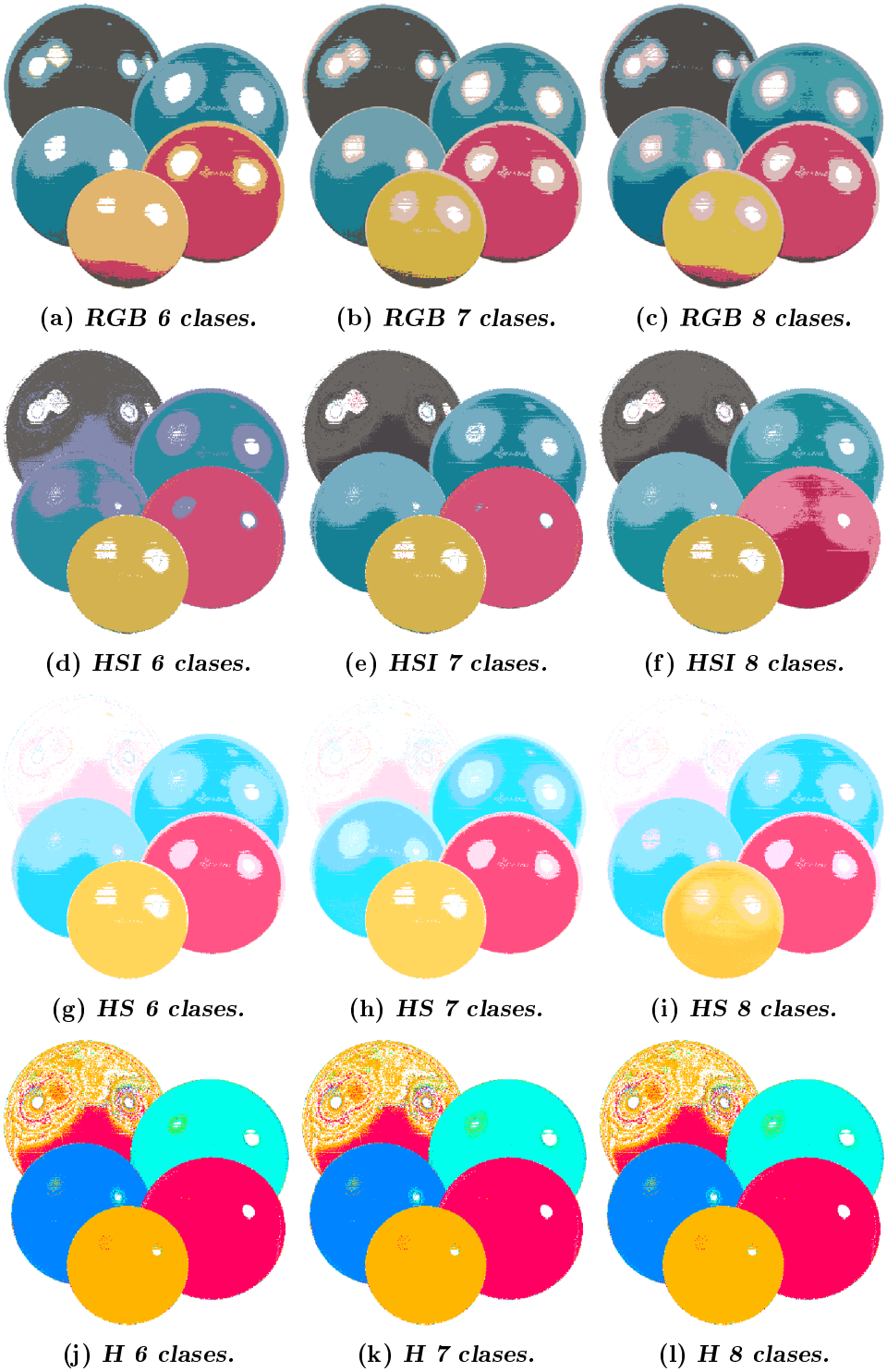
**Agrupamiento (Clustering).** El agrupamiento, o "clustering" como se conoce en la literatura, es un método de clasificación no supervisado en el que hay que decidir las clases o particiones del espacio a clasificar sin un conocimiento a priori. Es un método de clasificación usado en múltiples aplicaciones, entre ellas la segmentación en color como se puede ver en [Lucchese01].

La principal ventaja del "clustering" en la segmentación en color es que evita la determinación empírica de los umbrales de segmentación y, además, dependiendo del método usado, se puede decidir de manera automática el número de colores de la imagen a segmentar.

Las distintas variaciones sobre este método de segmentación usan distintos métodos de "clustering" (K-means o ISODATA por ejemplo) y distintos espacios de color. El método K-means es el más simple y a la vez efectivo para realizar el clustering, sin embargo, hay que decidir a priori el número de clases a obtener, o en nuestro caso, el número de colores a segmentar. El método ISODATA (Iterative Self-Organizing Data Analysis Technique) permite obtener de manera automática el número de colores de la imagen, para ello hay que definir a priori los parámetros de varianza máxima de la clase (a partir de la cual la clase se parte en dos) y distancia entre clases (para poder unir dos clases próximas) este proceso es iterativo hasta que se llega a un punto de equilibrio. El método ISODATA es más lento pero también obtiene mejores resultados en la segmentación. El uso de un espacio de color u otro, y dentro de un espacio, el uso de unas componentes u otras depende del tipo de aplicación que se vaya a realizar y de los objetivos a obtener.

En la figura 4.11 pueden verse varios ejemplos del uso de K-means en los espacios RGB y HSI. En este útlimo caso se muestran los resultados usando todas las componentes del espacio (Figuras 4.11(d), 4.11(e), 4.11(f)), sólo las componentes de tono y saturación (Figuras 4.11(g), 4.11(h), 4.11(i)) y sólo la componente de tono (Figuras 4.11(j), 4.11(k), 4.11(l)). Se observa cómo el uso del espacio RGB produce una sobre-segmentación debido a los cambios de iluminación y además las zonas de color próximas no se separan, como en el caso de las bolas azul y verde. Usando el tono y la saturación los resultados mejoran en el aspecto de la sobre-segmentación pero se puede ver cómo las zonas acromáticas (zonas con saturación baja) no son separables y, por tanto, el blanco y el negro están mezclados. También se aprecia el efecto de inestabilidad de la zona negra ya que ésta se separa en múltiples clases en vez de agruparse en una sola. Por último, el uso únicamente del tono produce los mejores resultados en cuanto a separación de colores pero sigue apareciendo el problema de la inestabilidad, que en este caso, aparece más marcado. Además hay que resaltar que el aumento del número de clases apenas afecta a la separación de las zonas de color lo que hace más estable este método y con menos tendencia a la sobre-segmentación.

En la figura 4.12 se muestran varios ejemplos del uso de ISODATA en los espacios RGB y CIE-Lab. En este caso se ha elegido el espacio CIE-Lab porque está especialmente diseñado para hacer que las distancias en el espacio de color sean equivalentes a las percibidas por el observador humano y, puesto que tanto K-means como ISODATA realizan la clasificación mediante el cálculo de distancias, estas sean proporcionales a la percepción



(a) ***RGB 6 clases.*** (b) ***RGB 7 clases.*** (c) ***RGB 8 clases.***

(d) ***HSI 6 clases.*** (e) ***HSI 7 clases.*** (f) ***HSI 8 clases.***

(g) ***HS 6 clases.*** (h) ***HS 7 clases.*** (i) ***HS 8 clases.***

(j) ***H 6 clases.*** (k) ***H 7 clases.*** (l) ***H 8 clases.***

Figura 4.11: Ejemplos de segmentación usando K-means. Cada zona de color de la imagen original está representada por un color parecido en la imagen segmentada, agrupando con dicho color a todos los que pertenecen a la misma clase.

humana. Los resultados de este método de "clustering" dependen de varios parámetros elegidos a priori, en particular de la varianza permitida dentro de las clases.

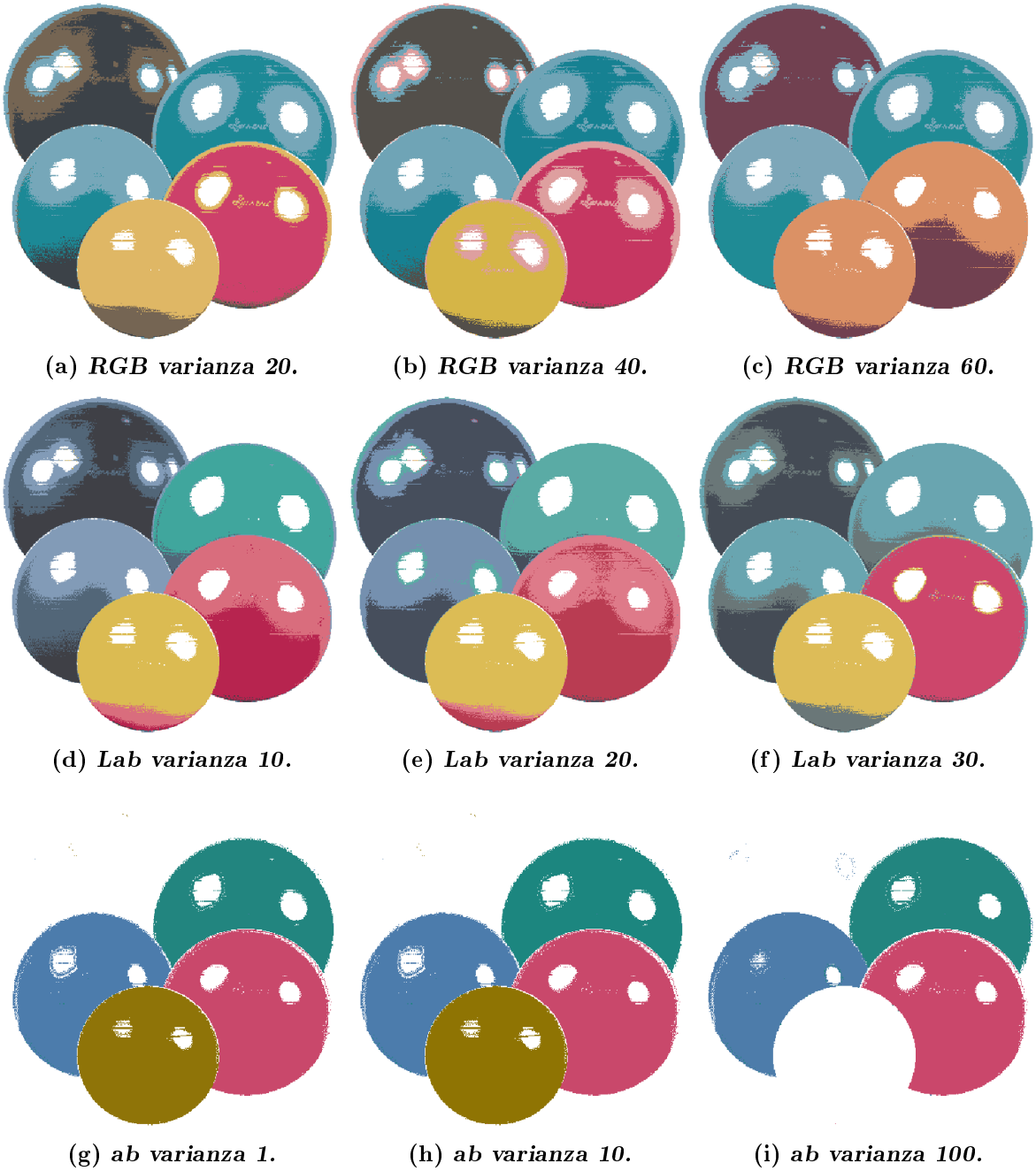
En las figuras 4.12(a), 4.12(b) y 4.12(c) se ha aplicado la segmentación en el espacio RGB con distintas varianzas. Se puede ver como el aumento de la varianza disminuye el número de clases (en este caso colores) dentro de la imagen aunque el método consigue separar de forma automática los elementos de color de la imagen. Al igual que en K-means, el uso del espacio RGB genera una sobre-segmentación de la imagen por depender estas componentes de la iluminación.

En las figuras 4.12(d), 4.12(e) y 4.12(f) se ha utilizado el espacio CIE-Lab con todas sus componentes y puede verse como los resultados también dependen de la varianza, reduciendo el número de colores con el aumento de la misma. Hay que destacar que la sobre-segmentación sigue apareciendo puesto que se está usando la componente de luminancia pero la separación de colores mejora ya que la pelota verde aparece separada de la azul para valores bajos de varianza. Si eliminamos la componente de luminancia se obtienen los resultados de las figuras 4.12(g), 4.12(h) y 4.12(i) donde se pueden apreciar dos hechos significativos, primero que las zonas blanca y negra no son separables puesto que en este espacio tienen las mismas componentes "ab" y segundo que la segmentación de las zonas no acromáticas es casi perfecta, separando completamente las cuatro bolas con colores definidos independientemente de las distintas zonas de iluminación de las mismas. En cuanto a la dependencia con la varianza, ésta es menor que en los otros casos, teniendo que aumentarla mucho para reducir el número de colores dentro de la imagen, como puede verse en la figura 4.12(i).

Cuando estos métodos de "clustering" se quieren utilizar para segmentar un color en concreto el proceso a seguir implica realizar la segmentación de una imagen de entrenamiento, la identificación del centroide asociado a ese color y posteriormente la asignación de los píxeles cercanos a ese centroide como del color requerido. Aunque la fase de entrenamiento puede ser costosa, sobre todo en ISODATA, para la posterior segmentación sólo es necesaria la medida de distancias y la asignación de clases. El proceso de entrenamiento debe ser repetido varias veces hasta obtener los resultados deseados porque la naturaleza aleatoria del inicio del clustering hace que los resultados cambien de un intento al otro aunque, como se ha comentado, esto no es un problema cuando se realiza el proceso de segmentación.

**Dividir y unir (Split&Merge).** Los métodos que se han presentado hasta ahora pertenecen a los que trabajan en algún espacio de características (espacio de color en nuestro caso), y clasifican los píxeles de la imagen únicamente según esa característica independientemente de la posición que puedan ocupar dentro de la imagen. Sin embargo, existen aplicaciones en las que la relación espacial es necesaria como característica para la segmentación, de manera que las zonas obtenidas sean homogéneas en color pero también en posición.

Dentro de las técnicas de segmentación en el dominio de la imagen se va a presentar la conocida como "Split&Merge" o Dividir y Unir en castellano. Esta técnica usa dos fases para realizar la clasificación de los píxeles de una imagen aunque cada una de las dos fases de por sí ya es una técnica de segmentación:

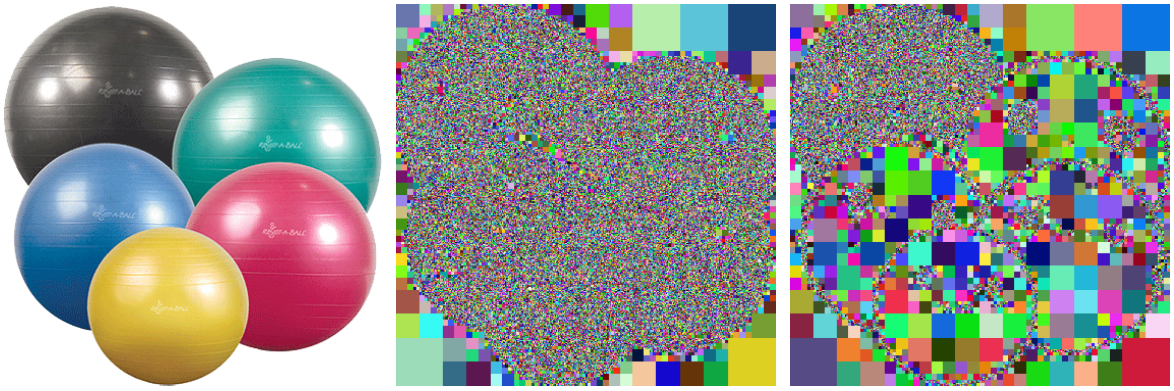


(a) ***RGB varianza 20.*** (b) ***RGB varianza 40.*** (c) ***RGB varianza 60.***

(d) ***Lab varianza 10.*** (e) ***Lab varianza 20.*** (f) ***Lab varianza 30.***

(g) ***ab varianza 1.*** (h) ***ab varianza 10.*** (i) ***ab varianza 100.***

Figura 4.12: Ejemplos de segmentación usando ISODATA. Cada zona de color de la imagen original está representada por un color parecido en la imagen segmentada, agrupando con dicho color a todos los que pertenecen a la misma clase.



**(a)** **(b)** **(c)**

Figura 4.13: Ejemplo de Split usando la diferencia entre máximo y mínimo como criterio de homogeneidad. (a) Imagen original. (b) Diferencia $\leq 0$. (c) Diferencia $\leq 10$.

1. **Dividir (Split)**. Se trata de un proceso recursivo en el que inicialmente todos los píxeles de la imagen se asignan a la misma clase. Si la imagen no cumple unos requisitos de homogeneidad (los píxeles que la forman son homogéneos en alguna característica), se divide en cuatro subimágenes del mismo tamaño a las que se les asigna cuatro clases distintas. Este proceso se repite para cada subimagen hasta que las subimágenes obtenidas sí cumplen el criterio de homogeneidad o ya no se puede dividir más. El criterio de homogeneidad puede ser variado, por ejemplo, que la varianza sea inferior a un determinado umbral o que la diferencia entre los valores máximo y mínimo sea pequeña. La aplicación de esta técnica nos da una imagen en la que las zonas espacialmente homogéneas aparecen marcadas como de la misma clase pero, en la práctica no se puede usar como técnica de segmentación. En la figura 4.13 se puede ver un ejemplo usando el espacio de color HSI con su componente de tono únicamente. Se aprecia como las zonas espacialmente homogéneas se marcan con el mismo color y cómo la relajación de los criterios de homogeneidad hace que las zonas sean más grandes. También se aprecia la dificultad de usarlo como técnica de segmentación.

2. **Unir (Merge)**. Este proceso es inverso al mostrado anteriormente. Se parte de una imagen en la que los píxeles han sido etiquetados de alguna manera y lo que se hace es comparar zonas adyacentes de la imagen para ver si cumplen unos criterios de similitud y, si se cumplen, esas dos zonas se unen y se les asigna la misma etiqueta. Esto se repite hasta que no existen más uniones dentro de la imagen. Un posible criterio de similitud es que las medias de dos zonas sean parecidas (su diferencia sea menor a un umbral). La clave de este método está en la elección del etiquetado inicial. Una posibilidad es el uso de la técnica de "Split" anterior para después unir las zonas que sean similares, aunque también se puede utilizar una segmentación inicial que no tenga homogeneidad espacial (Umbralizado o clustering, por ejemplo), que haya sido sobre-segmentada, para después unir las zonas homogéneas. Un ejemplo de ambos casos puede verse en la figura 4.14.

En este apartado se ha visto un ejemplo de las técnicas de segmentación en el dominio de la imagen que producen zonas segmentadas con coherencia espacial, lo que puede

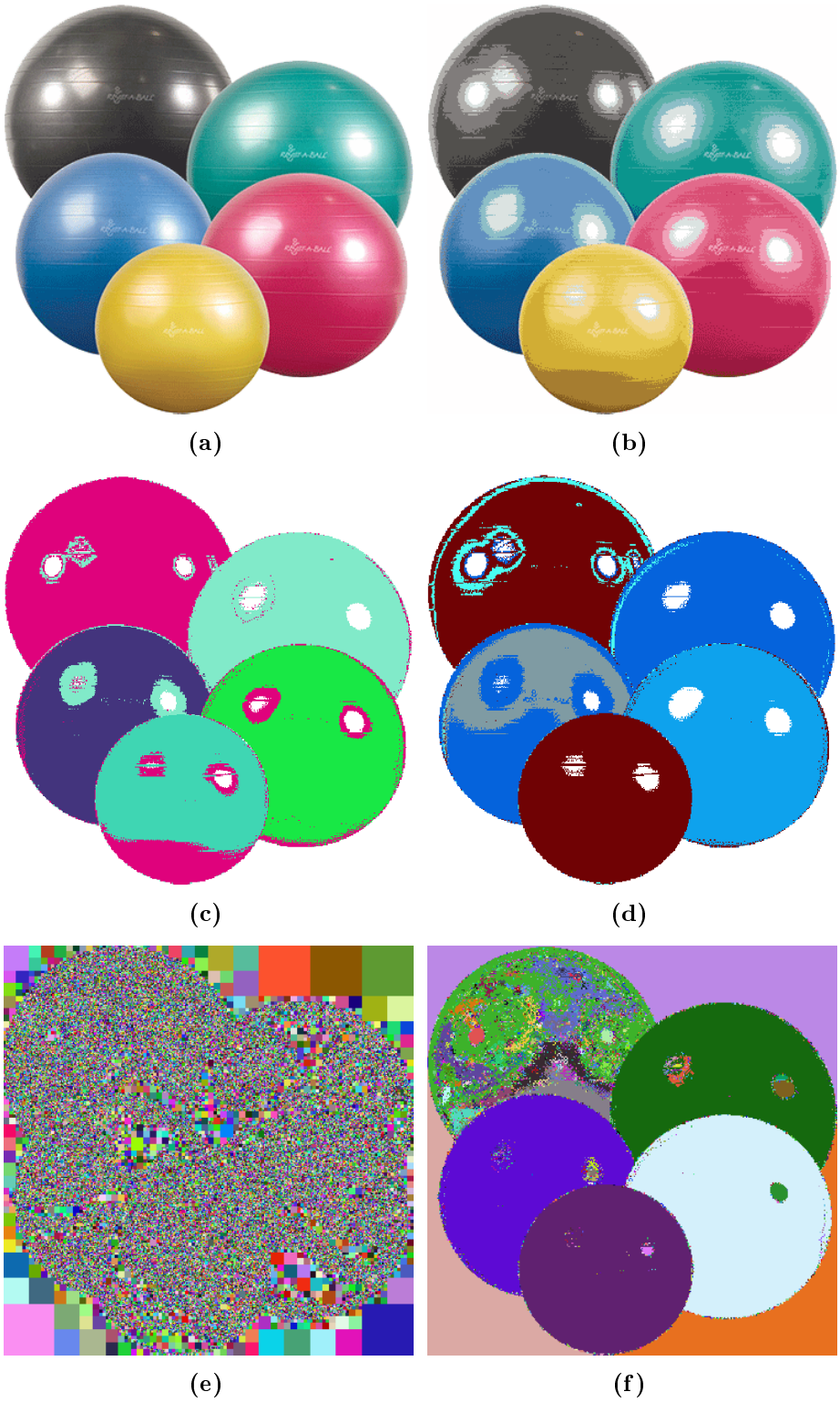



Figura 4.14: Ejemplo de la técnica Split-Merge. (a) Imagen original. (b) Imagen segmentada con K-means usando 30 clases. (c) Imagen obtenida partiendo de la imagen en (b) usando un criterio de similitud $\leq 15$. (d) Imagen obtenida partiendo de la imagen en (b) usando un criterio de similitud $\leq 20$. (e) Split inicial. (f) Imagen unión partiendo de la imagen en (e) y usando un criterio de similitud $\leq 15$.

ser útil para evitar la sobre-segmentación. Sin embargo, su principal defecto es su coste computacional ya que implican el uso de métodos iterativos o recursivos. Además, no es trivial identificar un determinado color con estos métodos puesto que las etiquetas cambian de una imagen a otra. Por tanto, su uso está limitado a aplicaciones en las que aparezcan texturas además de colores como característica de segmentación. También sería una opción de postprocesado el uso de la técnica de Unión después de la segmentación para reducir la sobre-segmentación, aunque a costa del aumento del tiempo de proceso.

## 4.2. Segmentación en color usando las SVM

En las secciones precedentes hemos tenido la oportunidad de estudiar algunas de las técnicas más importantes a la hora de realizar la segmentación de imágenes en color. En este apartado se propone el uso de las SVM para realizar esa tarea, esperando que las buenas cualidades de generalización y fácil entrenamiento, que ya se han explorado en esta herramienta, permitan facilitar la tarea de segmentación.

La tarea asignada a la SVM será la de clasificar los píxeles de una imagen entre aquellos que pertenecen a un determinado color y aquellos que no. Para ello deberemos generar de alguna manera el entrenamiento adecuado y ajustar los distintos parámetros de entrenamiento (Kernel, C, etc.). Este será el punto crucial para el éxito de este método.

En las secciones siguientes se pormenoriza la tarea de entrenamiento, la aplicación de la segmentación a distintas imágenes para comprobar su funcionamiento y la experimentación en distintos espacios de color. En una sección posterior se comparará el método presentado con otros de la literatura en una tarea en la que se necesita la segmentación, concretamente en la detección y reconocimiento de señales de tráfico.

### 4.2.1. Entrenamiento

Debido al buen funcionamiento conseguido en la tarea de detección de bordes mediante el uso de entrenamiento sintético, se pensó en su uso también para la tarea de segmentación en color. Sin embargo, se desechó esta posibilidad por la gran dificultad que supone generar distintos colores pertenecientes a un mismo tipo y realizar su etiquetado de manera automática, puesto que la percepción del color es básicamente subjetiva y está muy influenciada por las condiciones de iluminación y el entorno del píxel a considerar. Esa vertiente subjetiva aconsejó la obtención de los colores de entrenamiento mediante el etiquetado manual por un operador que eligiera las zonas de color deseadas. Para ello se diseñó una herramienta software que permite la extracción de zonas de una imagen para su posterior etiquetado y de esa manera facilitar la tarea.

Al realizar pruebas de entrenamiento se apreció enseguida una de las ventajas de este tipo de segmentación puesto que si el entrenamiento realizado no es el correcto o deseado en un primer momento sólo hay que añadir más muestras y repetir para ir consiguiendo los resultados deseados. En este caso es importante que las imágenes de entrenamiento y de test sean distintas para que el entrenamiento sea significativo.

**Ejemplo del proceso de entrenamiento.** Para ilustrar el proceso de entrenamiento, en este apartado se realizará un ejemplo de segmentación sobre una imagen de ejemplo.


**(a)** **(b)**

Figura 4.15: Ejemplo de extracción de colores en una imagen. (a) Extracción del color a detectar (b) Extracción de colores a descartar.

El primer paso será la extracción de muestras de la imagen del color a detectar y de los colores a descartar. En el ejemplo que se está desarrollando se utilizaron sólo dos bloques de píxeles mostrados en la figura 4.15. Una vez extraídos los píxeles y generado el fichero de patrones para el entrenamiento se han de elegir los parámetros adecuados de entrenamiento y realizar las pruebas necesarias. En este ejemplo se utilizará el espacio de color RGB y por tanto cada píxel estará representado por un vector de 3 componentes con valores entre 0 y 255. Para comprobar si con los píxeles elegidos se puede conseguir marcar la pelota de color verde se entrenará eligiendo el kernel RBF con un valor de C=1000 y un valor de $\gamma = 10^{-3}$. Con estos primeros resultados obtenidos se podrán ajustar posteriormente los parámetros de entrenamiento.

El número de vectores de entrenamiento en esos dos bloques es de 621, de los cuales 357 son de la clase +1 (color a detectar) y el resto de la clase −1 (colores a descartar). Con los parámetros expuestos inicialmente se obtienen 11 vectores soporte y un resultado de segmentación que es el mostrado en la figura 4.16(a). Puede comprobarse que efectivamente se logra marcar la bola deseada pero también parte de la azul contigua. Por tanto, se modifican los parámetros haciendo el valor de $\gamma = 10^{-4}$ y se obtienen 4 vectores soporte con el resultado mostrado en la figura 4.16(b). Es decir, haciendo más pequeño el valor de $\gamma$ se consigue generalizar más y ampliar los colores que entran dentro de una clase. Por tanto, hay que variar el valor de $\gamma$ en sentido contrario. La siguiente prueba se realizará con $\gamma = 5 \cdot 10^{-3}$, que es el valor intermedio entre los anteriores, siendo en este caso el número de vectores de 33 y el resultado el de la figura 4.16(c). Este resultado es más cercano a lo deseado pero se puede ir variando el valor de $\gamma$ hasta conseguir que se acerque al ideal. Para ello se modifica el valor hasta llegar a que con $\gamma = 2 \cdot 10^{-3}$ se obtuvieran 15 vectores y el resultado de la figura 4.16(d). Con este experimento se ha podido comprobar el efecto de $\gamma$ en la segmentación.

El efecto de $C$ en este caso es irrelevante porque los colores están muy bien separados y no existen vectores no separables. En otros casos más problemáticos sí que influye, pero básicamente en un reducción del número de vectores soporte si es grande. Por ello se decidió dejarlo fijo en $C = 1000$ para todos los casos.

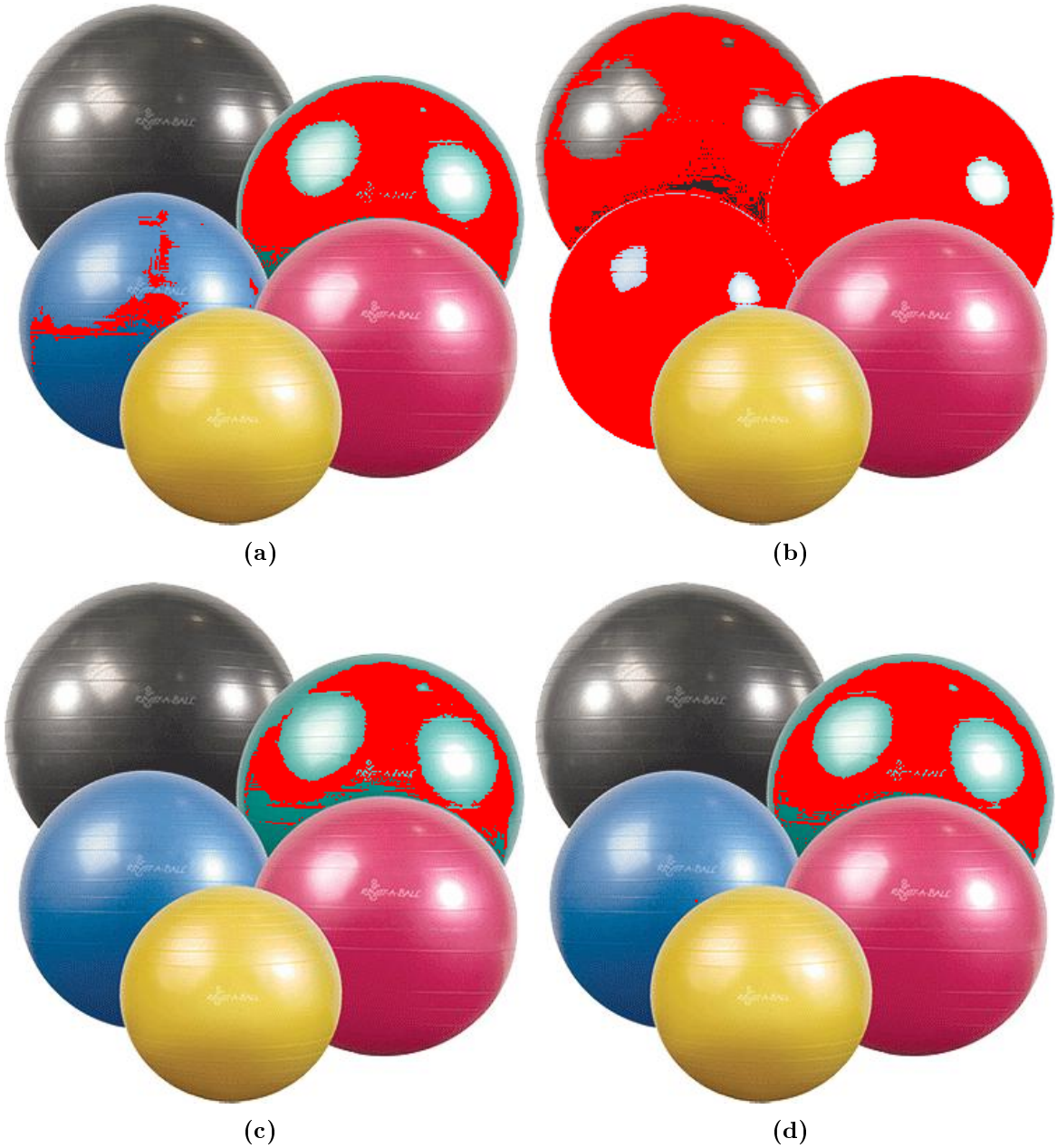



Figura 4.16: Distintos resultados de segmentación variando el entrenamiento. (a) $\gamma = 10^{-3}$, (b) $\gamma = 10^{-4}$, (c) $\gamma = 5 \cdot 10^{-3}$, (d) $\gamma = 2 \cdot 10^{-3}$

También se realizaron pruebas con otros kernel (erbf, polinomial, etc.) y con otras estrategias de minimización del error (L1-SVM, L2-SVM) pero los resultados o eran similares o peores. En el mejor de los casos el resultado era igual pero el número de vectores aumentaba. Por ejemplo, usando la estrategia L1-SVM se obtienen 15 vectores para $\gamma = 2 \cdot 10^{-3}$ y con L2-SVM 33.

Por todo lo expuesto anteriormente se decidió el uso del kernel Gaussiano en el resto de experimentos con la estrategia L1-SVM.

**Detección de varios colores.** Con los datos obtenidos en el experimento anterior ya se puede plantear la detección de varios colores sin más que realizar un entrenamiento para cada color a detectar, o bien, utilizar la multiclasificación asignando distintas etiquetas a distintos colores.

En primer lugar se implementará la opción de entrenamiento separado para cada color. Por tanto se extraerán píxeles correspondientes al color a detectar junto con otros a descartar que permitan la correcta segmentación. Para este experimento se usará las misma imagen de ejemplo del apartado anterior buscando segmentar cada una de las bolas por separado. En la figura 4.17 aparecen los resultados correspondientes a este tipo de segmentación. Puede comprobarse que los resultados son aceptables aunque hay zonas de la bola negra que se segmentan en amarillo y viceversa. Una nueva toma de vectores para eliminar esas zonas mejoraría la segmentación a costa quizás de perder otras zonas bien segmentadas.

Para ilustrar la capacidad de discriminación de las SVM en este caso, en la figura 4.18 se muestran los colores correspondientes a los vectores soporte obtenidos en cada clasificador. Cada franja de color corresponde con un vector soporte, los positivos a la izquierda y los negativos a las derecha. En el entrenamiento se eligieron primero las zonas a detectar y después se fueron añadiendo vectores de otros colores para "refinar", eso se refleja en el número de vectores soporte negativos pues su número (número de franjas) es mayor que el de positivos. El número de vectores soporte para cada color se puede consultar en la tabla 4.1, en concreto, para el espacio $RGB$.

Otra posibilidad para la segmentación es el uso de multiclasificación. En ese caso los vectores de cada color se etiquetan con un número distinto y se realiza el entrenamiento buscando los vectores soporte correspondientes a cada clase. Para este experimento se tomaron los vectores de entrenamiento ya utilizados para la prueba anterior aunque en este caso sólo se usaron los etiquetados con $+1$ para cada color. Con esos vectores se formó un nuevo conjunto de entrenamiento con etiquetas distintas para cada color. Además se le añadieron vectores de color blanco del fondo para tener todas las clases presentes en la imagen de prueba. Esto se hizo porque si no se incluye ese color, sus píxeles son clasificados como amarillos por ser la clase más próxima. En total se obtuvieron 3680 vectores. Para el entrenamiento se usaron los mismos parámetros que en el caso anterior y se obtuvieron 86 vectores soporte. La técnica utilizada para la multiclasificación fue la de 1 contra 1 y votación final.

La figura 4.19 muestra el resultado obtenido sobre la imagen de prueba. Nuevamente el resultado obtenido es satisfactorio aunque para el caso del color amarillo la segmentación se extiende a zonas que no le corresponden. Lo que sí se puede apreciar es una reducción en el número de vectores soporte pasando de un total de 133 para la suma de todos los clasificadores a los 86 obtenidos en mutliclasificación. En la figura 4.20 se presentan los

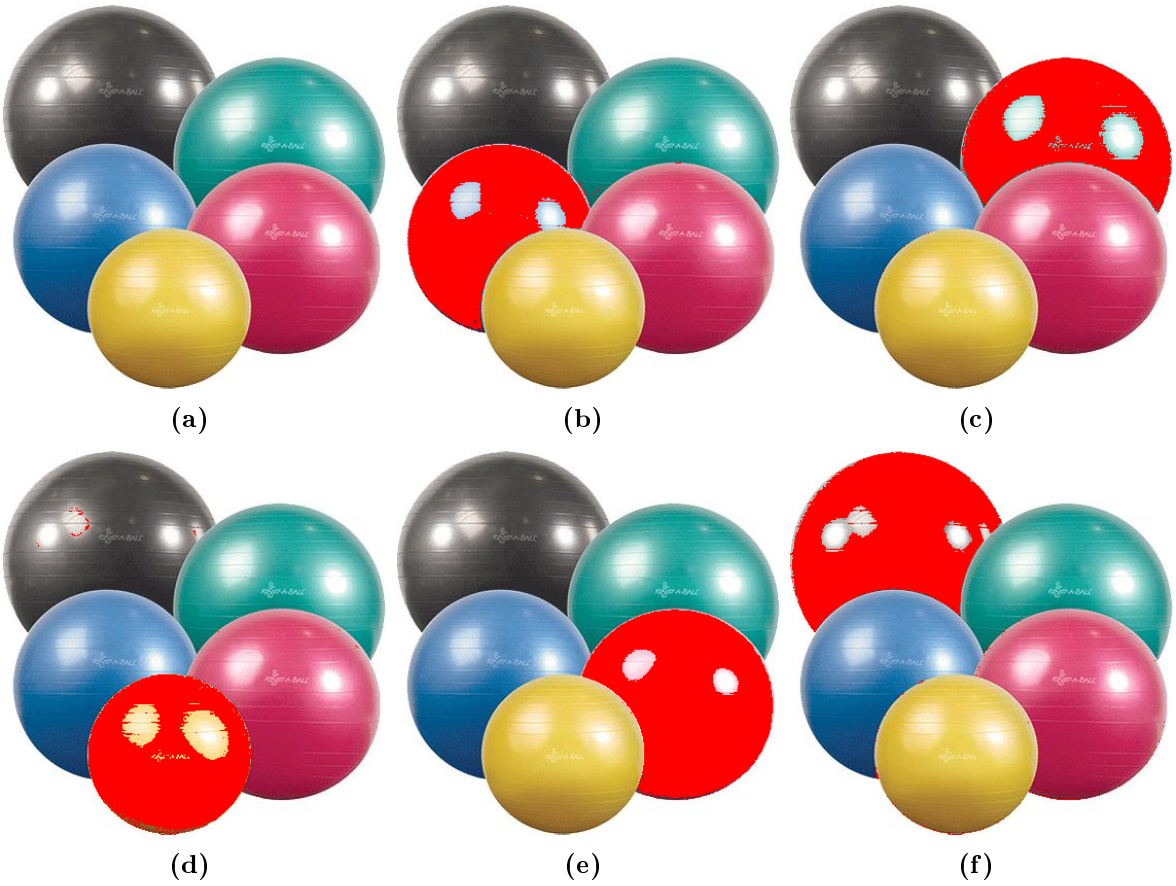



Figura 4.17: Resultados de segmentación obtenidos usando el espacio *RGB* y SVM. Parámetros $\sigma = 4 \cdot 10^{-4}$ y $C = 1000$.

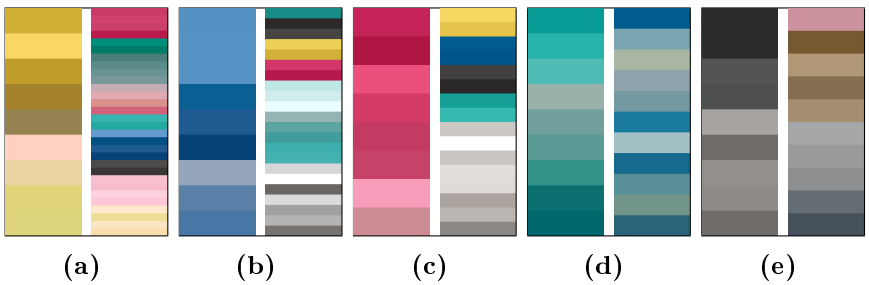



Figura 4.18: Representación gráfica de los vectores soporte obtenidos para cada clasificador. (a) Amarillo, (b) Azul, (c) Rojo, (d) Verde, (e) Negro.

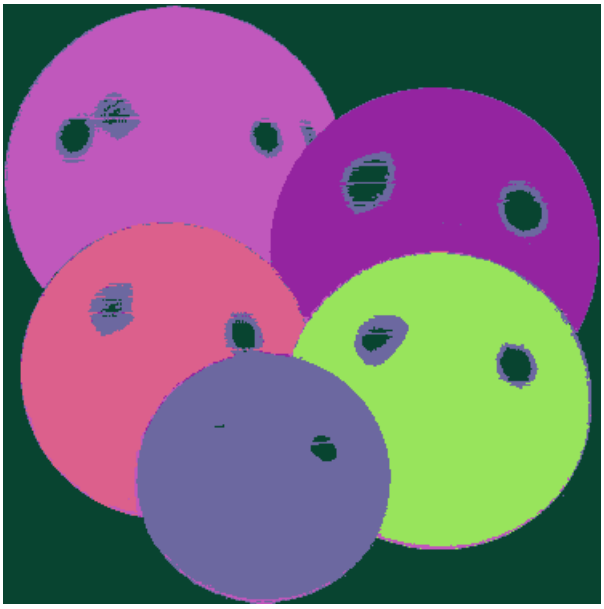


Figura 4.19: Resultado de la segmentación usando multiclasificación.

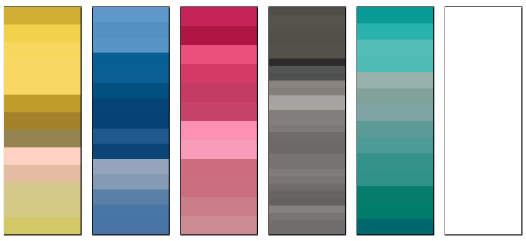



Figura 4.20: Representación gráfica de los vectores soporte obtenidos con el multiclasificador. (a) Amarillo, (b) Azul, (c) Rojo, (d) Verde, (e) Negro y (f) Blanco.

colores correspondientes a los vectores soporte de cada clase al igual que se hizo en la prueba anterior.

Como conclusión final se puede decir que ambos modos de segmentación son válidos y tienen distintos tipos de aplicación. Cuando el objetivo sea extraer zonas de ciertos colores de la imagen la opción sería el uso de distintas SVM (una para cada color). Si se quiere segmentar por colores la imagen completa debería usarse multiclasificación etiquetando todos los colores posibles. El uso de multiclasificación para sólo unos pocos colores tiene el riesgo de que lo no etiquetado sea también clasificado y los resultados sean erróneos.

### 4.2.2. Experimentación en varios espacios de color

Una vez establecido el correcto funcionamiento del método propuesto en el espacio de color RGB queda pendiente la comprobación del efecto de usar otros espacios en esta misma tarea. En la sección 4.1.1 se estudiaron distintos espacios de color desde el punto de vista de la segmentación y, basándonos en los datos obtenidos en ella, se han elegido como posibles candidatos los espacios *rgb*, *HSI* y *Ohta*. El resto de espacios o bien tienen

Tabla 4.1: Número de vectores soporte para cada color y cada espacio de color.

| Color \ Espacio | RGB | rgb | HSI | Ohta |
|---|---|---|---|---|
| Amarillo | 39 | 37 | 40 | 13 |
| Azul | 31 | 9 | 45 | 17 |
| Rojo | 24 | 7 | 21 | 16 |
| Verde | 20 | 13 | 26 | 11 |
| Negro | 19 | 147 | 49 | 13 |

transformaciones complicadas y costosas ($L^*a^*b^*$ o $L^*u^*v^*$) o bien no específicas para segmentación ($YUV$, $YIQ$) o no aportan ninguna mejora en cuanto a la segmentación sobre $RGB$, como el $XYZ$.

En la figura 4.21 se muestran los resultados obtenidos con esos espacios y su comparación con los ya presentados para la segmentación en el espacio $RGB$. Además, en la tabla 4.1 se muestra el número de vectores soporte correspondiente a cada espacio de color y para cada una de las máquinas entrenadas en función del color a detectar. Hay que aclarar que en todos los espacios las componentes han sido escaladas para estar en el rango $[0, 255]$ de forma que los parámetros de entrenamiento ($C$ y $\gamma$) puedan ser de la misma magnitud. Además esos parámetros se han ajustado al mismo valor ($C = 1000, \sigma = 4 \cdot 10^{-4}$) porque los resultados obtenidos resultaron ser los mejores en todos los casos.

Observando los resultados obtenidos podemos destacar las siguientes observaciones:

1. El peor de los espacios en cuanto a resultados es el $HSI$ pues se puede comprobar como tanto en el caso del amarillo, como en el del azul se marcan más zonas de las debidas. Esto mismo se produce para el rojo pero en menor medida. Se aprecia también la inestabilidad de este espacio con colores acromáticos como es el caso del negro donde el marcado no es homogéneo. Además, el número de vectores soporte es relativamente alto.

2. El espacio $rgb$ sí da buenos resultados de segmentación para la mayoría de los colores y además tiene un número de vectores reducido. Sin embargo, se aprecia también la inestabilidad respecto a los colores con intensidad baja como, en este caso, el negro donde además de no dar buenos resultados de segmentación el número de vectores es anormalmente alto.

3. El espacio de color *Ohta* también da buenos resultados de segmentación (comparables a $RGB$) pero con una reducción en el número de vectores soporte. Como además la transformación correspondiente a este espacio es sencilla podemos tenerlo en cuenta a la hora de buscar una alternativa a $RGB$.

Con los resultados que se acaban de mostrar parece que, salvo el espacio *Ohta*, el uso de espacios de color distintos a $RGB$ no producen mejoras importantes. Sin embargo, para la obtención de esos resultados se mantuvo el uso de todas las componentes propias de los espacios de color analizados, sin tener en cuenta que una posibilidad es usar sólo

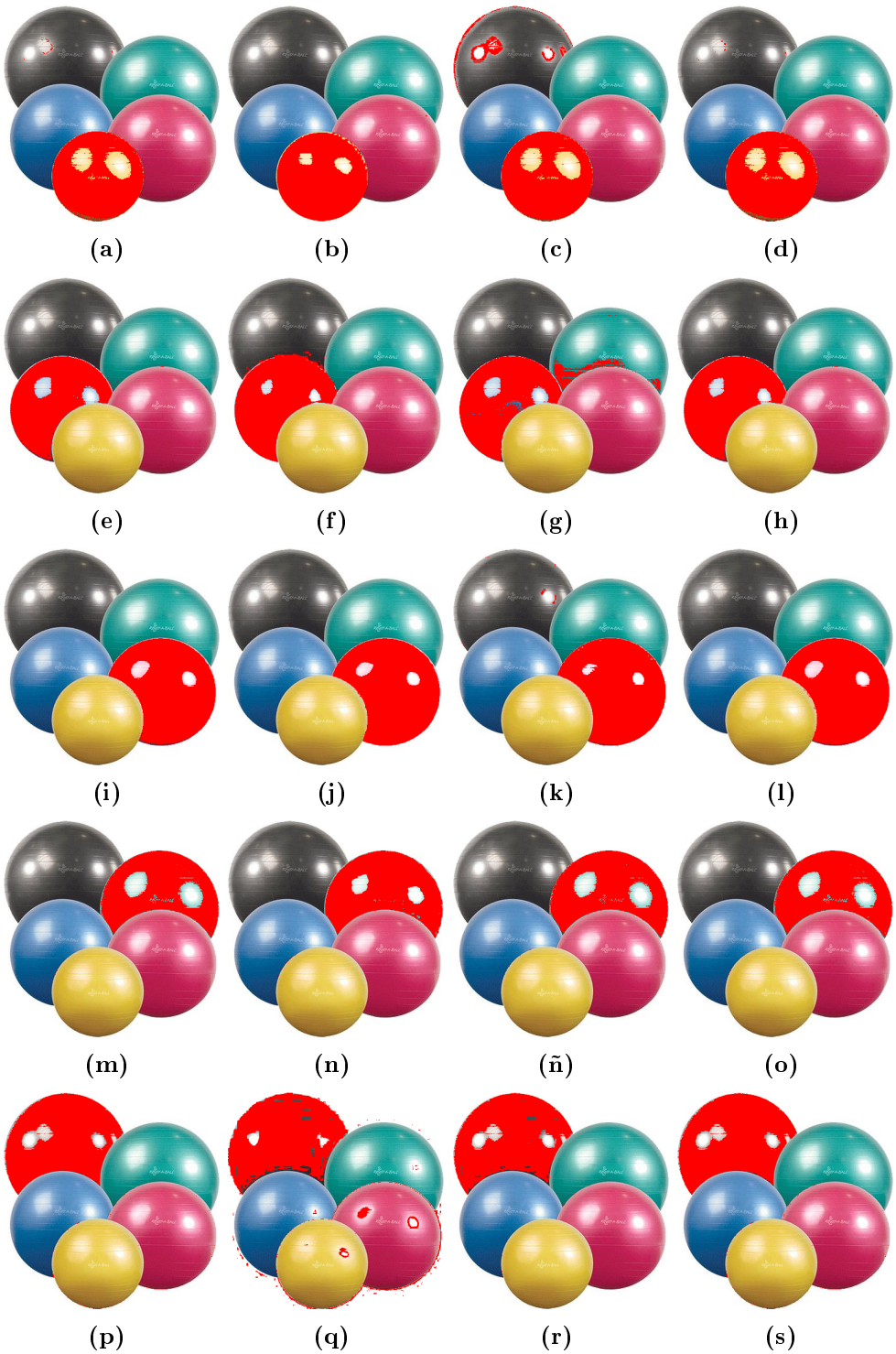



Figura 4.21: Resultados de segmentación para distintos espacios de color. De izquierda a derecha *RGB*, *rgb*, *HSI*, *Ohta*. De arriba hacia abajo colores Amarillo, Azul, Rojo, Verde y Negro

Tabla 4.2: Número de vectores soporte para cada color y cada espacio de color reducido.

| Color \ Espacio | rgb | HSI | Ohta |
|---|---|---|---|
| Amarillo | 44 | 36 | 58 |
| Azul | 8 | 27 | 6 |
| Rojo | 5 | 7 | 6 |
| Verde | 14 | 19 | 11 |
| Negro | 188 | 55 | 180 |

aquellas componentes que están directamente relacionadas con la información cromática. En función del espacio del que se trate las componentes a utilizar serían:

- En el espacio $HSI$ pueden utilizarse sólo las componentes $H$ (Tono) y $S$ (Saturación).
- Según se explico en 4.1.1 en $rgb$ sólo usando dos componentes tenemos toda la información necesaria. Por ello para el estudio siguiente se utilizaron las componentes $r$ y $g$.
- En el espacio $Ohta$ la componente $I_1$ es de iluminación pues representa la media de las componentes $RGB$. Así que la información de color estaría en las componentes $I_2^*$ e $I_3^*$.

Con esa reducción de componentes, a priori, se reduciría el tiempo de ejecución y la complejidad de la segmentación. Todo dependerá de si se mantienen los resultados y el número de vectores soporte. En la figura 4.22 se muestran los resultados correspondientes a los espacios de color reducidos y en la tabla 4.2 el número de vectores soporte correspondientes.

En la obtención de los resultados presentados se han mantenido los parámetros de entrenamiento, para poder comparar la versión total y la reducida de los espacios de color. Para dichos resultados podemos comentar lo siguiente:

1. Una vez más los peores resultados de segmentación se dan en el espacio $HSI$. Incluso para el color amarillo se incluye mucha parte blanca que antes se discriminaba gracias a la componente de intensidad. Además se aumenta la zona segmentada de color negro. Esto, nuevamente, es debido a la pérdida de la componente de intensidad que aumenta la confusión entre los distintos colores acromáticos. En cuanto al número de vectores soporte destaca el hecho de que para los colores más puros (Rojo, azul y verde) este número se reduce de manera importante mientras que se mantiene en el caso del amarillo.

2. Para el espacio $rgb$ el resultado de segmentación es similar y también se produce la reducción de vectores soporte antes comentada para $HSI$. Sin embargo, el problema con el color negro se agudiza en resultados y número de vectores soporte.

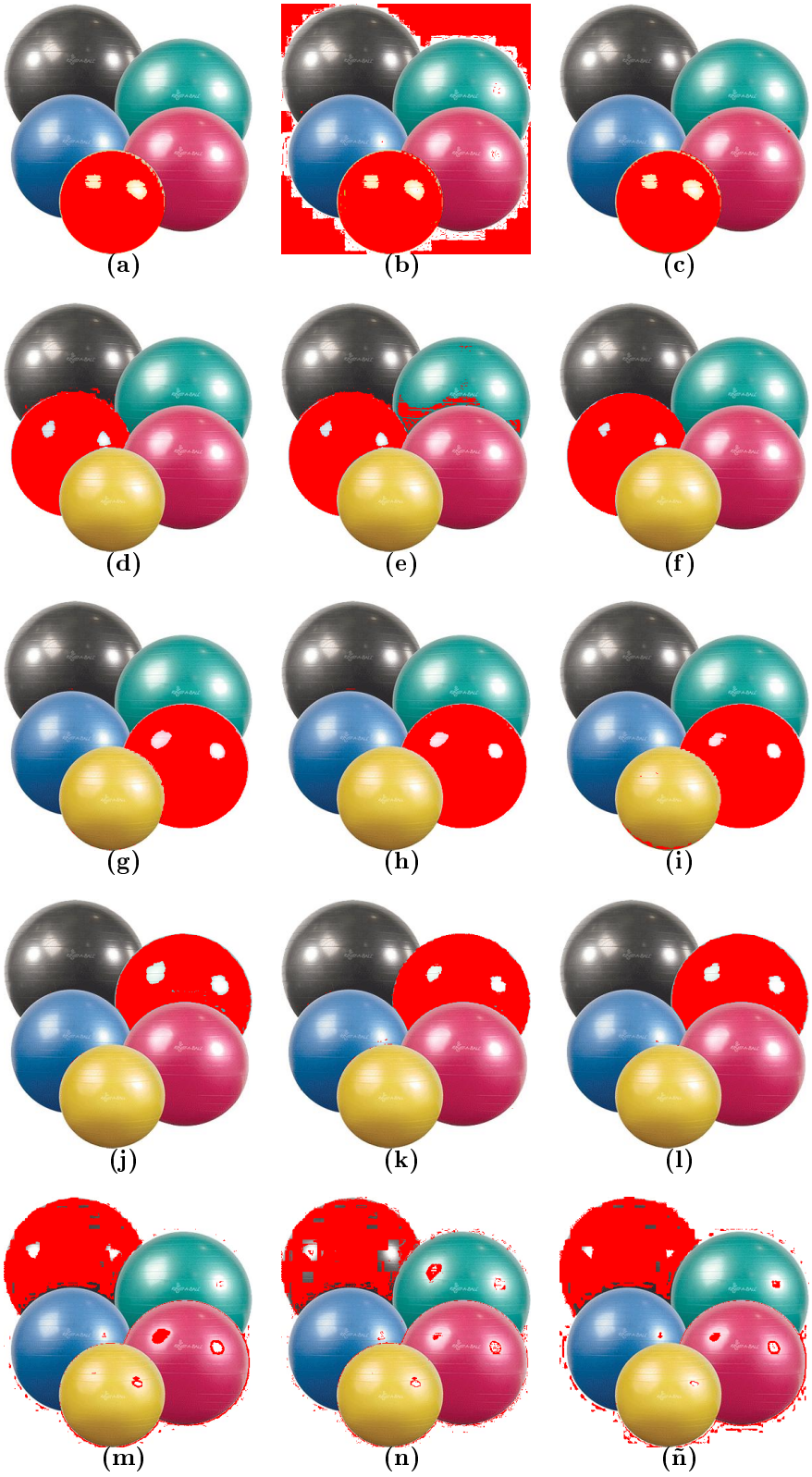



Figura 4.22: Resultados de segmentación usando espacios de color reducidos. De izquierda a derecha *rgb*, *HSI* y *Ohta*. De arriba hacia abajo colores Amarillo, Azul, Rojo, Verde y Negro.

3. En el espacio *Ohta* los resultados de segmentación se mantienen salvo para el negro. Sin embargo, la reducción de vectores sólo se produce en el caso de los colores más puros. Para el amarillo y el negro se produce un aumento, más acusado en este último.

Como conclusión final, se puede afirmar que el uso de otros espacios de color puede ser adecuado para reducir el número de vectores soporte si se quieren detectar colores más puros. Como opción para reemplazar al espacio *RGB* se puede usar el *Ohta* con todas sus componentes porque produce resultados similares pero con menos vectores soporte.

## 4.3. Evaluación y presentación de resultados

Los resultados presentados hasta este momento han sido de tipo visual pero no se ha podido determinar la calidad del método propuesto en comparación con otros de la literatura de una forma objetiva.

Como ya se comentó en el capítulo dedicado a la detección de bordes, aunque la literatura al respecto propone un número significativo de métodos para la medida de la calidad de un algoritmo de segmentación ([Yang95, Zhang96, Zhang01a, Chabrier04b]), ninguno de ellos ha sido aceptado como estándar. Estos métodos sufren de dos problemas fundamentales, o bien necesitan un "ground-thruth" para comparar (y este es normalmente generado a mano) o bien calculan una serie de estadísticos que son específicos para un determinado escenario. En ambos casos el tiempo de ejecución es excesivo si el número de imágenes es grande.

Por otro lado hay que tener en cuenta que la segmentación es, en sí misma, una tarea subjetiva. Por ejemplo, en [Heath97] se utilizaron distintos sujetos entrenados para determinar la calidad de la segmentación, sin embargo, cada uno de ellos puntuaba de manera distinta la misma imagen segmentada. Es decir, distintas personas marcarán las zonas de segmentación de distinta manera y, además, en el caso del color se sabe que la percepción del mismo no sólo cambia con el individuo sino también con otros factores como la iluminación. Al tratar en este caso con un algoritmo de segmentación y no con uno de detección de bordes, se optó por buscar alternativas al uso de los índices de calidad de la literatura.

Siguiendo la línea mostrada por ejemplo en [Zhang01a], muchos investigadores tienden a evaluar la calidad de la segmentación desde el punto de vista de la aplicación en la que va a usarse. Así, en esta tesis se propone un método de evaluación en el que la segmentación formará parte de un sistema de reconocimiento automático de señales de tráfico y, por tanto, la calidad de la segmentación se reflejará en el mejor o peor reconocimiento de dichas señales. La idea es aplicar el sistema de reconocimiento variando únicamente el método de segmentación usado y tratar los resultados últimos obtenidos en el proceso de reconocimiento como parámetro de calidad de la segmentación. Los resultados y las conclusiones obtenidas con este método han sido publicadas recientemente en [Gómez-Moreno10].

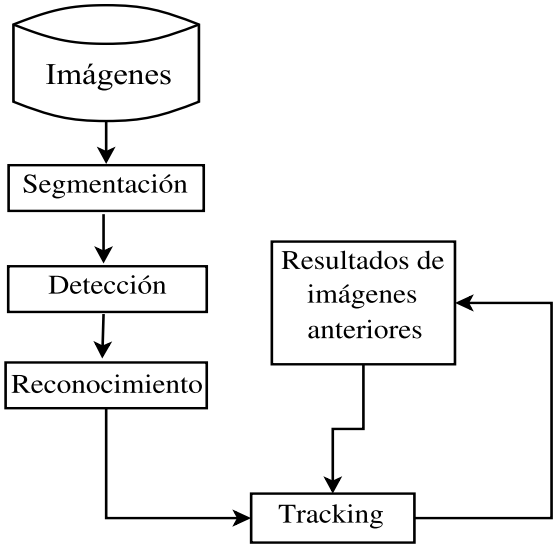



Figura 4.23: Diagrama de bloques de un sistema de reconocimiento de señales de tráfico.

### 4.3.1. El sistema de reconocimiento de señales de tráfico.

El sistema de reconocimiento de señales de tráfico en el que se basa la medida de calidad fue presentado y descrito en detalle en [Maldonado-Bascón07]. Este sistema consta de 4 bloques diferenciados (Figura 4.23):

- **Segmentación**: Este bloque separa los objetos del fondo, en este caso señales de tráfico usando la información de color.
- **Detección**: En este bloque las potenciales señales de tráfico son localizadas y se clasifican en función de su forma.
- **Reconocimiento**: Llegados a este punto, y en función de la forma y el color, se identifica la señal dentro de las posibles opciones. Este bloque se realiza mediante SVM.
- **Tracking**: En este bloque se agrupan los sucesivos reconocimientos que, se supone, corresponden a la misma señal.

Estos bloques han sido analizados con amplitud en [Gil-Jiménez08, Maldonado-Bascón07, Lafuente-Arroyo06] pero a continuación se realiza un pequeño resumen de los mismos, excepto de la segmentación que es el objeto de estudio de este capítulo de la tesis.

**Detección de formas.** La detección de formas que se presenta aquí fue propuesta en [Gil-Jiménez08]. Este paso usa la máscara de salida de la segmentación y da la posición y la forma dentro de las posibles de una señal de tráfico. Dentro de esa máscara los píxeles se agrupan por proximidad y color formando lo que en inglés se denomina "blob" que no tiene una traducción clara en castellano (podría ser algo así como mancha) y que usaremos en el resto del texto. Los blobs se pueden obtener mediante las distintas técnicas

de segmentación que se vieron en la sección 4.1 pero, principalmente, por umbralización, clasificación o marcando los bordes dentro de la imagen. Posteriormente se presentarán los métodos de segmentación elegidos.

Las formas que se consideran en este bloque son el triángulo, el círculo, el rectángulo y el semicírculo. El octógono (de las señales de Stop) se considera como un círculo. La detección se lleva a cabo comparando la firma de cada blob con aquellas obtenidas de formas de referencia. La firma no es más que la distancia al centro del blob puesta en función del ángulo.

Para reducir la distorsión de proyección, cada blob se normaliza usando el ángulo del eje de mínima inercia como referencia y la excentricidad se reduce usando un método basado en los momentos de orden 2. La oclusión se intenta reducir mediante la interpolación de la firma en las zonas en las que el blob está abierto.

En el caso de los problemas de escalado y rotación se utiliza la Trasnsformada Discreta de Fourier. La transformada es invariante frente a las rotaciones y la normalización se realiza en la energía de las muestras de la misma. La firma utilizada toma 64 ángulos del contorno de la señal y, puesto que se utiliza sólo el valor absoluto, la DFT tendrá la mitad puesto que el resto son repetidos. Estás muestras de la DFT se comparan con aquellas correspondientes a los modelos de referencia para obtener la categoría a la cual pertenece el blob.

Finalmente, el blob se reorienta a una posición de referencia para simplificar el proceso posterior de reconocimiento, excepto en el caso de los círculos, que no tienen punto de referencia.

**Reconocimiento.** Este bloque se describe en detalle en [Maldonado-Bascón07], aquí sólo se tratan los aspectos más importantes del mismo.

Una vez que los blobs han sido clasificados conforme a su forma y color se puede iniciar el proceso de reconocimiento. En lugar de usar el conjunto completo de señales, el reconocimiento se divide entre los diferentes colores y formas. De esta manera el conjunto total de 552 señales de tráfico (Tabla 4.3) se reduce a un máximo de 114 señales por tipo y, por tanto, se mejora la velocidad.

El entrenamiento y el test se llevan a cabo de acuerdo al color y la forma de cada región en cuestión. Por tanto, y para reducir la complejidad del problema, cada blob se compara sólo con aquellas señales que tienen el mismo color y forma.

La entrada al reconocimiento por cada posible señal es un bloque de $31 \times 31$ píxeles en escala de grises y que está normalizado en tamaño. El interior del bloque a reconocer es, por tanto, normalizado a esa dimensión y sólo los píxeles de interés son tenidos en cuenta para la clasificación dependiendo de la forma.

Se crean y entrenan diferentes SVMs con la estrategia uno-contra-todos y que usan el kernel gaussiano como base para la clasificación. En la fase de test la clase de señales de tráfico con el valor más alto en su función de decisión se asigna al blob en cuestión.

**Tracking.** La fase de tracking, [Lafuente-Arroyo06], identifica correspondencias entre señales de tráfico previamente reconocidas para dar una única salida por cada señal que aparezca en la secuencia. Si una nueva señal es detectada y no se corresponde con alguna anterior se inicia un nuevo proceso de tracking. La estructura de datos del tracking se refresca con esa nueva información para tenerla en cuenta a partir de ese punto. Al

Tabla 4.3: Número de señales reconocidas en función del tipo

| **Tipo** | **Número** |
|---|---|
| Circulares Rojas | 61 |
| Triangulares Rojas | 45 |
| Circulares Blancas | 110 |
| Triangulares Blancas | 44 |
| Rectangulares Blancas | 114 |
| Rectangulares Azules | 97 |
| Circulares Amarillas | 47 |
| Triangulares Amarillas | 25 |
| Rectangulares Amarillas | 9 |
| **Total** | **552** |

menos dos detecciones de una misma señal se requieren para considerar reconocida una señal. La información de posición, tamaño, color, tipo y el valor medio de los valores de gris de la región ocupada por el objeto, son los parámetros que se evalúan para poder establecer las correspondencias entre señales de tráfico de distintas imágenes.

### 4.3.2. Métodos de segmentación a comparar

Esta sección describe los algoritmos de segmentación evaluados (Tabla 4.4). La implementación de esos algoritmos generan máscaras binarias en las que se separan los objetos del fondo. Se obtiene una máscara por cada color de interés, es decir, rojo, azul, amarillo y blanco. Sin embargo, algunos de los algoritmos no son capaces de obtener todas las máscaras necesarias, en ese caso la única máscara obtenida se usa para todos los colores de interés. Este es el caso de las técnicas de detección de bordes.

Llegados a este punto, y tras unas pruebas iniciales, se constata un problema con la segmentación del color blanco, ya que este no es un color cromático sino acromático. En algunos trabajos sobre detección de señales de tráfico esta información no se trata y sólo la parte de color de las señales con colores rojo y azul es tenida en cuenta. Sin embargo, de esta forma se pierde mucha información ya que muchas señales contienen el color blanco, por ejemplo las señales de prohibición o peligro. Además, algunos espacios de color, como el $HSI$, son inestables cerca de los colores acromáticos y no se pueden utilizar directamente en esos píxeles.

Para poder mejorar la segmentación por blanco y reducir los problemas de algunos espacios de color en esta tesis se propone una descomposición Cromático/Acromático. Por ello, sólo aquellos píxeles considerados como cromáticos se clasifican entre los distintos colores, mientras que para los acromáticos solo aquellos que están por encima de un determinado valor de intensidad se consideran como blancos. Esta idea, basada en los valores de Saturación e Intensidad se usó en [Plataniotis00] pero aquí se adapta a distintos espacios de color para identificar los píxeles acromáticos. Por ello, en esta tesis se distingue

Tabla 4.4: Métodos de segmentación comparados. Las abreviaturas correspondientes se han mantenido en inglés como en [Gómez-Moreno10].

| **MÉTODOS DE SEGMENTACIÓN** | | |
|---|---|---|
| **Acromáticos** | CAD | Índice Cromático/Acromático |
| | RGB | Diferencias en RGB |
| | RGBN | Diferencias en RGB Normalizado |
| | SI | Saturación e Intensidad |
| | Ohta | Componentes Ohta |
| **Umbralización en espacios de color** | RGBNT | Umbralización en RGB Normalizado |
| | HST | Umbralización de Tono y Saturación |
| | HSET | Umbralización de Tono y Saturación mejorados |
| | OST | Umbralización en el espacio Ohta |
| **Detección de bordes** | GER | Eliminación de bordes en escala de grises |
| | Canny | Eliminación de bordes Canny |
| | CER | Eliminación de bordes en color |
| **Otros** | SVMC | Segmentación con SVM en color |
| | LUT | Mejora de la velocidad con tablas de búsqueda |

entre algoritmos de segmentación en color y aquellos que sólo buscan la descomposición entre cromático y acromático, aunque en algunos casos están muy relacionados.

**Umbralizado en espacios de color.** Una técnica muy extendida en segmentación (como se pudo ver en la sección 4.1, página 88) es la de utilizar la umbralización en distintos espacios de color. Las distintas variaciones de la misma ([Cheng01]) cambian el espacio de color utilizado y la técnica para identificar los umbrales. La elección del espacio de color es crucial en esta técnica ([Kumar02]) y por ello, en esta tesis se comparan algunos de ellos.

Es posible obtener un umbral automático basándose en el histograma de la imagen ([Otsu79]) pero de esta manera se identifican todas las regiones de color dentro de la imagen y no un color en particular. Para extraer un determinado color los umbrales se pueden fijar mediante observación en el espacio de color usado (ver página 88). La distribución de colores da una idea de los umbrales que se necesitan y que, posteriormente, se fijan de manera empírica para obtener los mejores resultados.

La obtención empírica de los umbrales no garantiza los mejores resultados y para refinar y mejorarlos se propone una búsqueda exhaustiva alrededor de los umbrales empíricos. Este procedimiento y algunos resultados se muestran en la sección 4.3.5. Como se ha comentado anteriormente, la clasificación del color blanco se realizará mediante el uso de las técnicas de descomposición Cromático/Acromático que se presentarán en la página 117.

a) **Umbralizado rgb (RGBNT).** Como ya se ha visto en secciones anteriores, el espacio RGB es un espacio básico y el que se toma como espacio inicial pues es el que usan las cámaras y los monitores. Si lo que se busca es simplicidad es el espacio adecuado pues no necesita transformaciones. Sin embargo, existe una alta correlación entre sus componentes y los cambios de iluminación le afectan en la modificación de la información de color. Por ello la búsqueda de umbrales para segmentación se complica. Una posible solución sería el uso de su versión normalizada como en [Kamada90, Janssen93]. De esta manera se reduce el impacto de la iluminación y además sólo son necesarios dos componentes para realizar la segmentación (ver sección 4.1.1). Sin embargo, hay que tener en cuenta la inestabilidad propia de este espacio para valores bajos de iluminación.

   Las máscaras para cada color en este espacio se obtienen usando las siguientes expresiones:

Tabla 4.5: Valores de los umbrales para los métodos RGBNT, HST y OST

| **Método** | **Valor de umbral** |
|---|---|
| RGBNT | $ThR = 0.4$, $ThG = 0.3$, $ThB = 0.4$, $ThY = 0.85$ |
| HST | $ThR_1 = 10$, $ThR_2 = 300$, $ThB_1 = 190$, $ThB_2 = 270$, $ThY_1 = 20$, $ThY_2 = 60$, $ThY_3 = 150$ |
| OST | $ThR_1 = 0.024$, $ThR_2 = -0.027$, $ThB_1 = -0.04$, $ThB_2 = 0.082$, $ThY_1 = 0.071$, $ThY_2 = 0.027$ |

$$\text{Rojo}(i,j) = \begin{cases} \text{Verdadero} & \text{si } r\,(i,j) \geq ThR \\ & \text{y } g(i,j) \leq ThG \\ \text{Falso} & \text{resto} \end{cases},$$

$$\text{Azul}(i,j) = \begin{cases} \text{Verdadero} & \text{si } b\,(i,j) \geq ThB \\ \text{Falso} & \text{resto} \end{cases}, \tag{4.13}$$

$$\text{Amarillo}(i,j) = \begin{cases} \text{Verdadero} & \text{si } (r(i,j) + g(i,j)) \geq ThY \\ \text{Falso} & \text{resto} \end{cases}.$$

Los valores de umbrales fijados aparecen en la Tabla 4.5.

**b) Umbralización del Tono y la Saturación (HST).** En [Lafuente-Arroyo04], se propuso una segmentación usando el tono y la saturación del espacio HSI. Como se ha visto en la sección 4.1.1 el tono representa el color dominante y la saturación la pureza del mismo con valores altos indicando colores puros y con bajos indicando alta mezcla de blanco. Los componentes del espacio HSI se pueden obtener a partir del RGB [Plataniotis00]. En el caso aquí presentado los valores de Tono estarán en el intervalo $[0, 360]$ y los de Saturación en $[0, 255]$. Se puede comprobar que el Tono está indefinido cuando la Saturación es nula (escala de grises $R = G = B$) y que la Saturación está indefinida cuando la Intensidad es nula.

La máscara de segmentación se puede obtener mediante umbralización en este espacio mediante las siguientes expresiones:

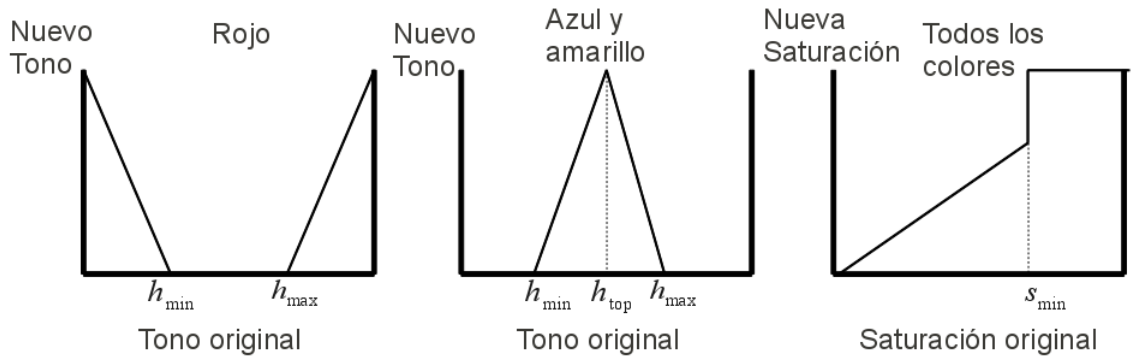



Figura 4.24: Transformaciones para el método HSET [delaEscalera04]. La mejora del color se consigue usando tres Tablas de búsqueda de las componentes de Tono y Saturación. Para el rojo se utiliza la tabla de la izquierda, el azul y el amarillo la central y la saturación la de la derecha. El Tono y la Saturación se han normalizado en el intervalo $[0, 255]$.

$$\text{Rojo}(i,j) = \begin{cases} \text{Verdadero} & \text{si } H(i,j) \leq ThR_1 \\ & \text{o } H(i,j) \geq ThR_2 \\ \text{Falso} & \text{resto} \end{cases},$$

$$\text{Azul}(i,j) = \begin{cases} \text{Verdadero} & \text{si } H(i,j) \geq ThB_1 \\ & \text{y } H(i,j) \leq ThB_2 \\ \text{Falso} & \text{resto} \end{cases}, \tag{4.14}$$

$$\text{Amarillo}(i,j) = \begin{cases} \text{Verdadero} & \text{si } H(i,j) \geq ThY_1 \\ & \text{y } H(i,j) \leq ThY_2 \\ & \text{y } S(i,j) \geq ThY_3 \\ \text{Falso} & \text{resto} \end{cases}.$$

En [Maldonado-Bascón07] los umbrales se fijaron después de un análisis de los histogramas del Tono y la Saturación de señales de tráfico seleccionadas manualmente. La tabla 4.5 muestra los umbrales empleados en esta tesis puesto que son diferentes a los presentados en dicho trabajo y además la Saturación se usa sólo en la detección del color amarillo.

Este método es simple y casi inmune a los cambios de iluminación ya que se usa el Tono, pero la inestabilidad del Tono y el incremento de tiempo debido a la transformación son sus mayores inconvenientes.

c) **Umbralización en Tono y Saturación mejorados (HSET).** En [delaEscalera04], se presenta un método diferente de umbralización para el Tono y la Saturación. Se estudia la distribución del Tono y la Saturación en señales rojas y azules segmentadas manualmente y se identifican los valores asociados a cada color. Para prevenir los problemas propios de los umbrales rígidos se propone un umbral suave basado en Tablas

de búsqueda que implementan la transformación de la figura 4.24. Este procedimiento es denominado "mejora del color", pero en realidad es un umbral suave, donde en vez de asignar el valor "1" a unos píxeles y "0" al resto se asignan diferentes valores usando una función lineal como las de la figura 4.24. Las tablas de transformación del Tono y la Saturación se describen en esa figura y en las ecuaciones 4.15, 4.16 y 4.17 donde son necesarios 4 parámetros: $h_{\min}$, $h_{\text{top}}$, $h_{\max}$ y $s_{\min}$.

La transformación para el color rojo es:

$$H_t(h) = \begin{cases} 255\frac{h_{\min 1}-h}{h_{\min 1}} & 0 \leq h \leq h_{\min 1} \\ 0 & h_{\min 1} < h < h_{\min 2} \\ 255\frac{h-h_{\min 2}}{255-h_{\min 2}} & h_{\min 2} \leq h \leq 255 \end{cases}, \tag{4.15}$$

y para el azul:

$$H_t(h) = \begin{cases} 0 & 0 \leq h \leq h_{\min} \\ 255\frac{h-h_{\min}}{h_{\text{top}}-h_{\min}} & h_{\min} < h < h_{\text{top}} \\ 255\frac{h_{\max}-h}{h_{\max}-h_{\text{top}}} & h_{\text{top}} < h < h_{\max} \\ 0 & h_{\max} \leq h \leq 255 \end{cases}. \tag{4.16}$$

Además para la Saturación será:

$$S_t(s) = \begin{cases} s & 0 \leq s \leq s_{\min} \\ 255 & s_{\min} \leq s \leq 255 \end{cases}. \tag{4.17}$$

En todos los casos el Tono y la Saturación están normalizados en el intervalo $[0, 255]$.

Los autores de [delaEscalera04] no incluyen el amarillo, por tanto en esta tesis se extiende dicho método al incorporar ese color usando el mismo método de transformación que para el azul (Fig. 4.24), pero utilizando distintos parámetros para la misma.

Finalmente, el método de segmentación se puede resumir como sigue:

1. Obtener el Tono y la Saturación de las componentes RGB de la imagen.
2. Transformarlos usando las tablas anteriormente mencionadas.
3. Normalizar el producto del Tono y la Saturación trasformadas. En esta tesis este paso no se realiza para aumentar la velocidad.
4. Umbralizar el producto final.

Los umbrales usados para cada color se muestran en la tabla 4.6. Estos valores se han obtenido modificando los propuestos en [delaEscalera04], ya que el producto no se normaliza y se adaptan para obtener los mejores resultados.

**d) Umbralización en el espacio Ohta (OST).**

El espacio de color de Ohta ya fue presentando previamente y se definieron sus propiedades siendo estas la simplicidad y la incorrelación de sus componentes. Además puede ser incluido en la familia de los espacios de color oponentes (Opponent Color Spaces) inspirados en la fisiología del sistema visual humano [Plataniotis00].

Tabla 4.6: Tabla de búsqueda y umbrales para el método HSET

| | $h_{\min}$ | $h_{\text{top}}$ | $h_{\max}$ | $s_{\min}$ | Umbrales |
|---|---|---|---|---|---|
| Rojo | 11 | N.A. | 224 | 23 | 28160 |
| Azul | 128 | 150 | 180 | 84 | 6257 |
| Amarillo | 10 | 20 | 25 | 84 | 14080 |

Recordemos que las componentes de este espacio en función de RGB serían:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 0 & -1 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{4.18}$$

Como se puede ver, la componente $I_1$ se relaciona con la iluminación y, por tanto, sólo $I_2$ e $I_3$ son necesarias para la clasificación del color. Aunque esas componentes se pueden usar directamente en segmentación, en este trabajo se usa la normalización presentada en [Vertan00] con la idea de reducir las variaciones debidas a los cambios de iluminación. Con ello los nuevos componentes $P_1$ y $P_2$ normalizados serían:

$$\begin{aligned} P_1 &= \frac{1}{\sqrt{2}} \frac{R-B}{R+G+B} = \frac{1}{3\sqrt{2}} \frac{I_2}{I_1}, \\ P_2 &= \frac{1}{\sqrt{6}} \frac{2G-R-B}{R+G+B} = \frac{2}{3\sqrt{6}} \frac{I_3}{I_1}. \end{aligned} \tag{4.19}$$

Con ellos la clasificación se realizaría:

$$\text{Rojo}(i,j) = \begin{cases} \text{Verdadero} & \text{si } P_1(i,j) \geq ThR_1 \\ & \text{y } P_2(i,j) \leq ThR_2 \\ \text{Falso} & \text{resto} \end{cases},$$

$$\text{Azul}(i,j) = \begin{cases} \text{Verdadero} & \text{si } P_1(i,j) \leq ThB_1 \\ & \text{y } |P_2(i,j)| \leq ThB_2 \\ \text{Falso} & \text{resto} \end{cases}, \tag{4.20}$$

$$\text{Amarillo}(i,j) = \begin{cases} \text{Verdadero} & \text{si } P_1(i,j) \geq ThY_1 \\ & \text{y } |P_2(i,j)| \leq ThY_2 \\ \text{Falso} & \text{resto} \end{cases}.$$

Los valores de los umbrales usados aparecen en la tabla 4.5.

**Descomposición Cromático/Acromático.** Mediante el uso de la descomposición Cromático/Acromático se trata de encontrar los píxeles que no contienen información de color, es decir, los píxeles grises. La escala de grises se obtiene cuando los valores de las componentes $R$, $G$ y $B$ también son iguales, sin embargo, si esos valores están próximos pero no son iguales los colores se perciben cercanos al gris. Todos los métodos que se van a presentar son diferentes entre sí ya que usan distintos espacios de color pero todos ellos se basan en la cercanía entre los valores de $R$, $G$ y $B$.

El resultado obtenido con estos métodos es que los píxeles grises se extraen y sólo aquellos cuya intensidad es grande se tratan como blancos mientras que el resto se descartan. Con esto, por una lado obtenemos los píxeles blancos y por otro eliminamos los píxeles acromáticos y de baja iluminación que son los más inestables en ciertos espacios de color.

a) **Índice Cromático/Acromático (CAD).** En [Liu02] se presentó un método para la descomposición Cromático/Acromático que se utilizaba para la detección de píxeles blancos. Ese método se usó posteriormente en [Maldonado-Bascón07] para detectar el blanco, junto con un umbralizado de Tono y Saturación. El método se basa en la definición de un índice Cromático/Acromático con la siguiente expresión:

$$\mathrm{CAD}(R,G,B) = \frac{(|R-G| + |G-B| + |B-R|)}{3D}, \tag{4.21}$$

donde $R$, $G$ y $B$ son las componentes de color para un determinado píxel y $D$ es el grado de extracción de un determinado color acromático. Por tanto, un píxel es considerado acromático cuando:

$$\mathrm{Acromático}(i,j) = \begin{cases} \text{Verdadero} & \text{si } \mathrm{CAD}(i,j) \leq 1 \\ \text{Falso} & \text{resto} \end{cases}, \tag{4.22}$$

y serán blancos cuando:

$$\mathrm{Blanco}(i,j) = \begin{cases} \text{Verdadero} & \text{si Acromático}(i,j) = \text{Verdadero} \\ & \text{y } (R+G+B) \geq ThW \\ \text{Falso} & \text{resto} \end{cases}. \tag{4.23}$$

Los valores fijados para este método se muestran en la tabla 4.7.

b) **Diferencias RGB.** Aunque el índice definido anteriormente es útil, es más realista si se usa un umbral diferente para medir la diferencia entre cada una de las componentes de de color. Por tanto, los colores acromáticos se pueden marcar cuando las tres diferencias entre los distintos componentes están por debajo de un umbral que es diferente para cada una. De esa manera un píxel se considera acromático cuando:

$$\mathrm{Acromático}(i,j) = \begin{cases} \text{Verdadero} & |R(i,j) - G(i,j)| \leq ThA_1 \text{ y} \\ & |G(i,j) - B(i,j)| \leq ThA_2 \text{ y} \\ & |B(i,j) - R(i,j)| \leq ThA_3 \\ \text{Falso} & \text{resto} \end{cases}, \tag{4.24}$$

Tabla 4.7: Umbrales para los métodos acromáticos

| **Método** | **Umbrales** |
|---|---|
| CAD | $D = 30$, $ThW = 180$ |
| Diferencias RGB | $ThA_1 = 32$, $ThA_2 = 40$, $ThA_3 = 40$, $ThW = 180$ |
| Diferencias RGB normalizado | $ThA = 0.17$, $ThW = 180$, $ThL = 60$ |
| SI Acromático | $ThA = 48$, $ThW = 60$, $ThL = 60$ |
| Ohta Acromático | $ThA_1 = 0.51$, $ThA_2 = 0.882$, $ThW = 60$, $ThL = 20$ |

y blanco cuando:

$$\text{Blanco}(i,j) = \begin{cases} \text{Verdadero} & \text{si Acromático}(i,j) = \text{Verdadero} \\ & \text{y } (R+G+B) \geq ThW \\ \text{Falso} & \text{resto} \end{cases} . \tag{4.25}$$

Los valores para los umbrales se muestran en la tabla 4.7.

**c) Diferencias en RGB Normalizado** En el espacio RGB normalizado los píxeles acromáticos se pueden etiquetar de una manera similar a la mostrada en el punto anterior. La diferencia es que en el espacio normalizado sólo son necesarias dos diferencias en lugar de las tres anteriores. Por tanto, la imagen de salida con los píxeles marcados como cromáticos o acromáticos se obtendría:

$$\text{Acromático}(i,j) = \begin{cases} \text{Verdadero} & \text{si } |r(i,j) - g(i,j)| \leq ThA \\ & \text{y } |r(i,j) - b(i,j)| \leq ThA \\ \text{Falso} & \text{resto} \end{cases} , \tag{4.26}$$

y blanco si se cumple:

$$\text{Blanco}(i,j) = \begin{cases} \text{Verdadero} & \text{si Acromático}(i,j) = \text{Verdadero} \\ & \text{y } (R+G+B) \geq ThW \\ \text{Falso} & \text{resto} \end{cases} . \tag{4.27}$$

Puesto que este es un espacio normalizado, con poca intensidad aparece una cierta inestabilidad y, para prevenirla, cuando la suma de las componentes RGB de un píxel considerado cromático está por debajo de un determinado umbral ($ThL$) es considerado directamente como negro y, por tanto, acromático. Los umbrales utilizados se muestran en la Tabla 4.7.

**d) Saturación e intensidad.** Cuando el espacio utilizado es el HSI se puede utilizar la detección Cromático/Acromático presentada en [Plataniotis00]. Este método se basa en el hecho de que valores bajos de Saturación indican píxeles acromáticos ya que para valores de $R$, $G$ y $B$ iguales la Saturación es nula. Sin embargo, no sólo los valores nulos se consideran acromáticos sino que valores bajos también lo son. Además, el Tono

está indefinido para Saturación nula y es inestable para valores bajos de intensidad. Finalmente, la expresión usada para la clasificación será:

$$\text{Acromático}(i,j) = \begin{cases} \text{Verdadero} & \text{si } S(i,j) \leq ThA \\ \text{Falso} & \text{resto} \end{cases} . \tag{4.28}$$

La intensidad debe considerarse de dos maneras. Los píxeles cromáticos pero con una Intensidad por debajo de un umbral ($ThL$) se consideran directamente como negros, previniendo la inestabilidad del Tono. Valores altos se considerarán blancos cuando el píxel sea acromático.

$$\text{Blanco}(i,j) = \begin{cases} \text{Verdadero} & \text{si Acromático}(i,j) = \text{Verdadero} \\ & \text{y } I(i,j) \geq ThW \\ \text{Falso} & \text{resto} \end{cases} . \tag{4.29}$$

Los valores de los umbrales se muestran en la Tabla 4.7.

**e) Componentes en el espacio Ohta** En el espacio Ohta los píxeles acromáticos pueden ser localizados examinando las componentes $I_2$ e $I_3$ o las normalizadas $P_1$ y $P_2$ (Ver ecuación 4.19). Valores bajos de $P_1$ y $P_2$ se obtienen cuando las componentes $R$, $G$ y $B$ son similares. Por ello, valores bajos de $P_1$ y $P_2$ marcan los píxeles acromáticos. De ese modo los píxeles acromáticos se marcarían usando:

$$\text{Acromático}(i,j) = \begin{cases} \text{Verdadero} & \text{si } |P_1(i,j)| \leq ThA_1, \\ & \text{y } |P_2(i,j)| \leq ThA_2, \\ \text{Falso} & \text{resto} \end{cases} , \tag{4.30}$$

y los blancos por:

$$\text{Blanco}(i,j) = \begin{cases} \text{Verdadero} & \text{si Acromático}(i,j) = \text{Verdadero} \\ & \text{y } I_1(i,j) \geq ThW \\ \text{Falso} & \text{resto} \end{cases} . \tag{4.31}$$

Para valores bajos de $I_1$ las componentes $P_1$ y $P_2$ son inestables, por ello los píxeles cromáticos con un valor de dicha componente por debajo de un umbral ($ThL$) se marcan como negros y, por tanto, acromáticos. Los valores usados para este método se muestran en la Tabla 4.7.

**Técnicas de Detección de bordes.** Otra forma de segmentación muy extendida es la basada en la detección de bordes para buscar los objetos dentro de una imagen. La idea es marcar los bordes como "no objeto" para que los puntos dentro de bordes cerrados se marquen automáticamente como "objeto". Con este método, la información de color no es necesaria y los problemas inherentes a los espacios de color se pueden prevenir.

En [Aoyagi96], los autores argumentan que el uso del color, y concretamente de la componente de Tono, tiene dos problemas fundamentales:

- Más costo computacional para obtener el Tono a partir del espacio RGB.
- La componente de Tono puede ser afectada por los cambios de iluminación, distancia a la cámara o las condiciones atmosféricas.

Por ello, proponen usar sólo el brillo de las imágenes para realizar la segmentación mediante el uso de la Laplaciana.

En [Garcia-Garrido06] los autores muestran que mientras que con el color es más rápido encontrar las zonas interesantes, la precisión es peor debido a: la confusión entre los colores (especialmente el rojo y el azul), los problemas de segmentación en las señales predominantemente blancas y a los cambios debidos a la iluminación. Por ello, los métodos basados en el análisis de formas son más robustos cuando hay cambios de iluminación. La opción es usar el método de Canny para detectar los bordes, ya que este método mantiene los bordes cerrados, una característica deseable en los sistemas de detección de formas.

Un problema común con los métodos de detección de bordes es que, aunque son simples y rápidos, producen numerosos objetos detectados, que cargan los procesos de detección y reconocimiento con más trabajo.

**a) Eliminación de bordes en escala de grises (Gray-Scale Edge Removal).** Este método se usó para la detección de señales de tráfico en [Aoyagi96] y comprende los dos pasos fundamentales del método de detección de bordes basado en la segunda derivada (ya presentado en 3.1.3, página 31) aunque con algunas variaciones:

- Suavizado de la imagen. Como la detección de bordes se basa en el filtro de Laplaciana, es conveniente realizar primero un suavizado de la imagen para reducir la amplificación del ruido. El filtro usado para realizar este suavizado se describe según la siguiente ecuación:

$$\begin{array}{l} g\left(i,j\right)=\{ \\ f\left(i-1,j-1\right)+f\left(i,j-1\right)+f\left(i+1,j-1\right)+ \\ f\left(i-1,j\right)+2f\left(i,j\right)+f\left(i+1,j\right)+ \\ f\left(i-1,j+1\right)+f\left(i,j+1\right)+f\left(i+1,j+1\right)\}/10 \end{array} ,$$

  donde $f$ es la imagen de entrada y $g$ la imagen suavizada.

- Aplicar el filtro de Laplaciana. El filtro de Laplaciana realiza la segunda derivada sobre la imagen para extraer los bordes. El filtro se describe en la siguiente ecuación:

$$\begin{array}{l} L\left(i,j\right)=g\left(i,j-1\right)+g\left(i-1,j\right)+ \\ g\left(i+1,j\right)+g\left(i,j+1\right)-4g\left(i,j\right) \end{array} ,$$

  donde $L$ es la imagen diferenciada y $g$ es la imagen previamente suavizada.

Partiendo de esto los bordes se obtendrán umbralizando la imagen resultado como sigue:

$$\mathrm{O}\left(i,j\right)=\begin{cases} \text{Borde} & \mathrm{L}\left(i,j\right)\geq T \\ \text{No-borde} & \mathrm{L}\left(i,j\right)<T \end{cases} , \tag{4.32}$$

siendo $T$ el umbral, que se ha fijado en $T=3$ como en [Aoyagi96].

**b) Eliminación de bordes Canny (Canny Edge Removal).** El método anterior es un detector muy simple y, aunque su implementación es rápida, su calidad no es la mejor. Entre los algoritmos de detección de bordes destaca el de Canny (presentado en 3.1.4, página 33) que es comúnmente reconocido ([Heath97]) como "estándar", siendo usado como comparación por muchos investigadores.

Como recordatorio, el método de Canny aplica un filtrado Gaussiano para suavizar, una derivación para obtener dirección y fuerza de los bordes en cada píxel y una "supresión de no-máximos" y umbralización con histéresis para marcar los bordes. Este método tiende a dar formas conectadas y los puntos aislados son mínimos.

Para este trabajo se usaron umbrales adaptativos basados en histogramas de la imagen. Los parámetros serán:

- *Sigma* o anchura del kernel Gaussiano.
- En vez de fijar un umbral superior se fija un valor de porcentaje de píxeles que han pasado la supresión de no-máximos. El umbral de magnitud se ajusta para que aplicado sobre el histograma se obtenga el número de píxeles deseado. El parámetro usado es *Thigh*.
- El umbral inferior se calcula como una fracción fija del umbral superior calculado anteriormente.

Los valores fijados para esos parámetros en este trabajo se muestran en la Tabla 4.8.

Tabla 4.8: Parámetros para la detección de Canny

| | *Sigma* | *Tlow* | *Thigh* |
|---|---|---|---|
| Valores | 0.5 | 0.5 | 0.95 |

**c) Eliminación de bordes en color (Color Edges Removal).** Los métodos anteriores no usan el color pero, considerando que puede ser una importante fuente de información, en este apartado se define un método de extracción de bordes en el espacio RGB. Este método mide la distancia entre un píxel y sus vecinos $3 \times 3$ en el espacio RGB. El proceso comienza con un detector de bordes aplicado sobre la imagen de color de acuerdo a la siguiente ecuación:

$$D_{ij} = \sum_{k=1}^{8} (R_{ij} - R_{ijk})^2 + (G_{ij} - G_{ijk})^2 + (B_{ij} - B_{ijk})^2, \tag{4.33}$$

donde $R_{ij}$, $G_{ij}$ y $B_{ij}$ son los valores rojo, verde y azul del píxel $ij$, y $R_{ijk}$, $G_{ijk}$ y $B_{ijk}$ son los valores de rojo, verde y azul del vecino $k$-ésimo. El valor que se obtiene no es directamente la distancia sino su cuadrado ya que con eso se reduce el tiempo de cómputo. Después de que se obtiene el valor $D_{ij}$ para cada píxel, aquellos con valores por debajo de un umbral se consideran pertenecientes al fondo mientras que los que están por encima pertenecen a bordes de la imagen que separan objetos.

En este método el umbral no se ha fijado a priori sino que se han realizado pruebas con distintos valores para ver los diferentes resultados.

**Segmentación basada en las máquinas de vectores soporte (Support Vector Machines Color Segmentation).** Los métodos de umbralización presentan dos problemas fundamentales. Hay un gran número de umbrales que ajustar y ese ajuste se hace sólo sobre unas determinadas imágenes, sin confirmación de que posteriormente se puedan generalizar a otras nuevas. Buscando reducir el trabajo en el ajuste de umbrales y que la generalización sea mayor se pensó en el uso de SVM para segmentar las imágenes en color como se presentó en la sección 4.2.

El espacio de color elegido ha sido el RGB por simplicidad. Los parámetros para las SVM se fijaron mediante búsqueda exhaustiva con las utilidades de ajuste de la librería LIBSVM [Chang01]. Finalmente los valores utilizados fueron, kernel RBF, $\gamma = 0.0004$ y $C = 1000$ para todos los colores.

Entre las ventajas de este método está el hecho de que se pueden buscar sólo los colores necesarios para nuestra aplicación de una forma sencilla. Para conseguir una buena generalización para distintas secuencias (diferentes cámaras o iluminaciones) el número de vectores de entrenamiento debe aumentar pero debido a la capacidad de generalización de las SVM el número de vectores no aumenta en la misma medida.

El principal problema es la velocidad. Aunque el número de vectores no es muy grande la velocidad es inferior a otros métodos de segmentación. Debido a ello no es conveniente su uso directo en el sistema de reconocimiento. Sin embargo, este problema se puede reducir si se usan tablas de búsqueda como las propuestas en el siguiente apartado.

**Mejora de la velocidad usando una tabla de búsqueda.** A veces, se dispone de un buen algoritmo de segmentación pero no se puede usar en una aplicación real porque es lento. Generalmente los algoritmos que requieren una transformación del espacio de color o cálculos complejos (como las SVM) presentan estos problemas de velocidad. Estos se pueden reducir con el uso de tablas de asignación precalculadas en las que a cada valor del vector RGB se le asigna un determinado color. Esa asignación se puede realizar con el algoritmo que se desee.

Con este esquema se necesita una tabla que contenga $2^{24}$ posiciones para conseguir una total equivalencia. El número de operaciones se reduce (únicamente se necesita un acceso a la tabla y una asignación) pero la tabla requerida es demasiado grande. Por eso, en las tablas definidas para este trabajo las componentes RGB se cuantifican con 6 bits en lugar de 8 y la tabla resultante tiene $2^{18}$ posiciones.

Esta cuantificación no debería afectar en gran manera a la segmentación porque sólo se eliminan los dos bits menos significativos de cada componente. La reducción en tiempo es lo bastante significativa como para no tener en cuenta esta pérdida de información.

En los experimentos llevados a cabo esta tabla se usa con tres métodos: el HST, que necesita el cálculo del tono y la saturación, el HSET, que calcula un tono y una saturación mejorados, y las SVM que requieren el cálculo de valores de kernel para realizar la clasificación. En el caso de las SVM las imágenes de entrenamiento han sido tomadas tanto en días lluviosos como soleados, con diferentes cámaras y distintas situaciones de iluminación diurna.

### 4.3.3. Definición del conjunto de pruebas

Para la validación y comparación de los métodos de segmentación presentados previamente es importante definir un conjunto de pruebas que sea lo suficientemente desafiante. Por ello, de las miles de imágenes que están disponibles debido al trabajo realizado por el grupo de investigación GRAM (Grupo de Reconocimiento y Análisis Multisensorial `http://agamenon.tsc.uah.es/Investigacion/gram`) se eligieron algunos subconjuntos, representativos de posibles dificultades de segmentación. La imágenes fueron tomadas en distintas rutas y con diferentes condiciones de iluminación. Las dificultades de segmentación que se fueron apreciando al probar el sistema en dichas secuencias son variadas e incluyen: baja iluminación, lluvia, agrupaciones de señales, color de fondo similar al de la señal, oclusiones, etc.

Esos conjuntos de imagen se unieron para formar el conjunto de **prueba inicial**. Posteriormente, y para validar los resultados obtenidos, se definió un nuevo conjunto llamado de **validación**. Finalmente, en un intento por comprobar el funcionamiento del sistema en cuanto al tracking, se usó una secuencia completa de imágenes de varios minutos tomada en distintos entornos, este es el conjunto de **tracking**.

A continuación se describen dichos conjuntos, indicando sus características y porqué se eligieron para las pruebas.

**Conjunto de pruebas inicial.** El conjunto de pruebas inicial está formado, a su vez, por 14 subconjuntos cada uno de ellos caracterizado por un problema de segmentación distinto. Un resumen de dichos subconjuntos se puede ver en la Tabla 4.9 donde se indican además sus principales características. En la figura 4.25 se muestran además las imágenes más significativas correspondientes a los subconjuntos estudiados.

El total de imágenes en este conjunto es de 313 siendo su tamaño de $800 \times 600$. Como se ha comentado, la selección se hizo pensando en su problemática respecto a la segmentación. El hecho de seleccionar un conjunto reducido viene condicionado porque un número grande imágenes no permite una inspección y etiquetado exhaustivo como el que necesita este estudio.

**Conjunto de validación.** Ya que los ajustes necesarios para los distintos métodos de segmentación se harán usando las imágenes del conjunto de pruebas inicial, se pensó en la creación de otro conjunto que sirva de validación a los resultados que se puedan obtener.

Para ello se usaron imágenes capturadas con distintas cámaras, bajo diferentes condiciones de iluminación y en diferentes localizaciones. Todos esos parámetros eran diferentes a los del conjunto inicial.

El conjunto de validación quedó formado por 43 señales distintas en 7 subconjuntos distintos y un total de 84 imágenes diferentes. En las figuras 4.26 y 4.27 se muestran las imágenes más representativas de este conjunto.

**Conjunto de tracking.** Con la idea de probar el comportamiento de cada método de segmentación usando el sistema completo, es decir incluyendo el tracking, se definió un nuevo conjunto de imágenes. En este caso se usó una secuencia de 7799 imágenes grabadas en un entorno mixto urbano y de carretera durante 12 kilómetros. Esta secuencia no tenía

Tabla 4.9: Conjuntos usados para probar los algoritmos de segmentación.

| **Nombre** | **Características** | **Número de Imágenes** | **Tipos de señales** |
|---|---|---|---|
| SET 1 | Colores y formas diferentes | 36 | 1:  2:  3:  4:  |
| SET 2 | Señales blancas complejas | 11 | 5:  |
| SET 3 | Colores y formas diferentes | 27 | 6:  7:  8:  |
| SET 4 | Señal blanca compleja | 9 | 9:  |
| SET 5 | Formas diferentes y un array de señales | 29 | 10:  11:  12:  13:  |
| SET 6 | Señales con fondo amarillo | 20 | 14:  15:  |
| SET 7 | Dos señales con un fondo rojizo y con tamaño pequeño comparadas con el de la imagen | 5 | 16:  |
| SET 8 | Array de señales con varias rojas cruzadas | 14 | 17:  18:  |
| SET 9 | Señales parcialmente ocluídas | 11 | 19:  |
| SET 10 | Presencia de arrays y fondo complejo. Día lluvioso | 10 | 20:  21:  |
| SET 11 | Presencia de arrays y fondo complejo. Día lluvioso | 17 | 22:  |
| SET 12 | Diferentes colores para segmentar y algunas señales rotadas | 27 | 23:  24:  25:  |
| SET 13 | Señales de color rojo cruzadas que son complejas de reconocer | 57 | 26:  27:  |
| SET 14 | Día nuboso con baja iluminación | 40 | 28:  29:  |


Figura 4.25: Algunas imágenes pertenecientes a los conjuntos de prueba. Estas imágenes son representativas de las señales que se pueden detectar en cada conjunto.


Figura 4.26: Algunas imágenes pertenecientes a los conjuntos de validación del 1 al 6. Estas imágenes son representativas de las señales que se pueden detectar en cada conjunto.

relación con las imágenes o secuencias usadas anteriormente. En la figura 4.28 se presentan, a modo de resumen, algunas imágenes correspondientes a dicha secuencia.

### 4.3.4. Comparación de los métodos.

Como ya se ha comentado, en esta tesis se propone un método de evaluación basado en el comportamiento de la segmentación como parte de un sistema completo de reconocimiento de señales de tráfico. La idea básica es contar las señales correctamente detectadas usando diferentes métodos de segmentación mientras que el resto de bloques del sistema (Detección y Reconocimiento) permanecen sin cambiar. Mediante inspección de las imágenes de prueba, se puede conocer el número y tipo de señales a reconocer y, por tanto, saber si se han reconocido todas las posibles. En todo caso este no es el único parámetro interesante a medir sino que otros como la velocidad o el número de señales perdidas también son importantes.

Los parámetros a evaluar serán entonces:

- **Número de reconocimientos por señal**. Para cada señal individual se cuentan las veces que ha sido correctamente reconocida[1]. Sin embargo, en lugar de mostrar el número de veces, se obtiene una versión normalizada respecto al máximo número de veces que es posible detectar dicha señal. De esta manera, un valor próximo a 1 sería el mejor escenario, ya que entonces todas las posibles señales se habrían reconocido.

- **Puntuación total**. Además del parámetro anterior (que es individual para cada señal), se presenta también la suma del número de señales reconocidas en todas los tipos de señales, para un mismo método de segmentación. De esta manera se obtiene una medida única para cada método. Ya que en el conjunto de test hay 29 tipos diferentes de señales la puntuación ideal sería de 29.

- **Porcentaje global de reconocimientos correctos**. La suma de todas las señales correctamente detectadas en todas las imágenes se hace relativa al número total de señales que pueden ser reconocidas globalmente. Esto nos da un porcentaje que indica el ratio de señales reconocidas sobre el total.

[1]Una señal se puede reconocer más de una vez por imagen si tiene varios colores. Por ejemplo, las señales circulares rojas tienen también un contenido blanco y por tanto se pueden detectar dos veces. Por ejemplo, la señal 4 del Set 1 de la figura 4.9

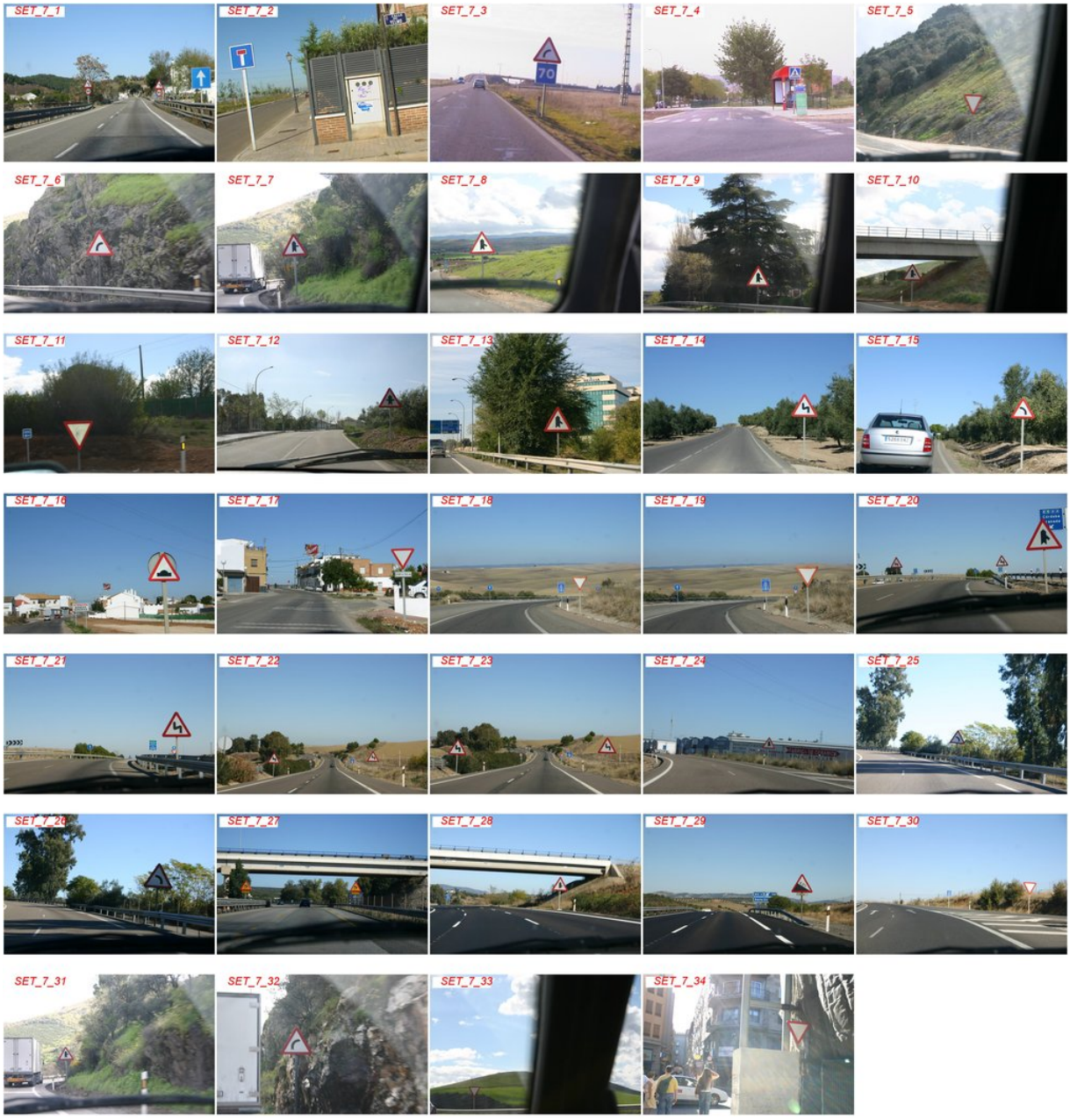



Figura 4.27: Imágenes pertenecientes al conjunto de validación 7. Este conjunto es una mezcla de imágenes pertenecientes a distintos momentos, carreteras y tipos de señales.

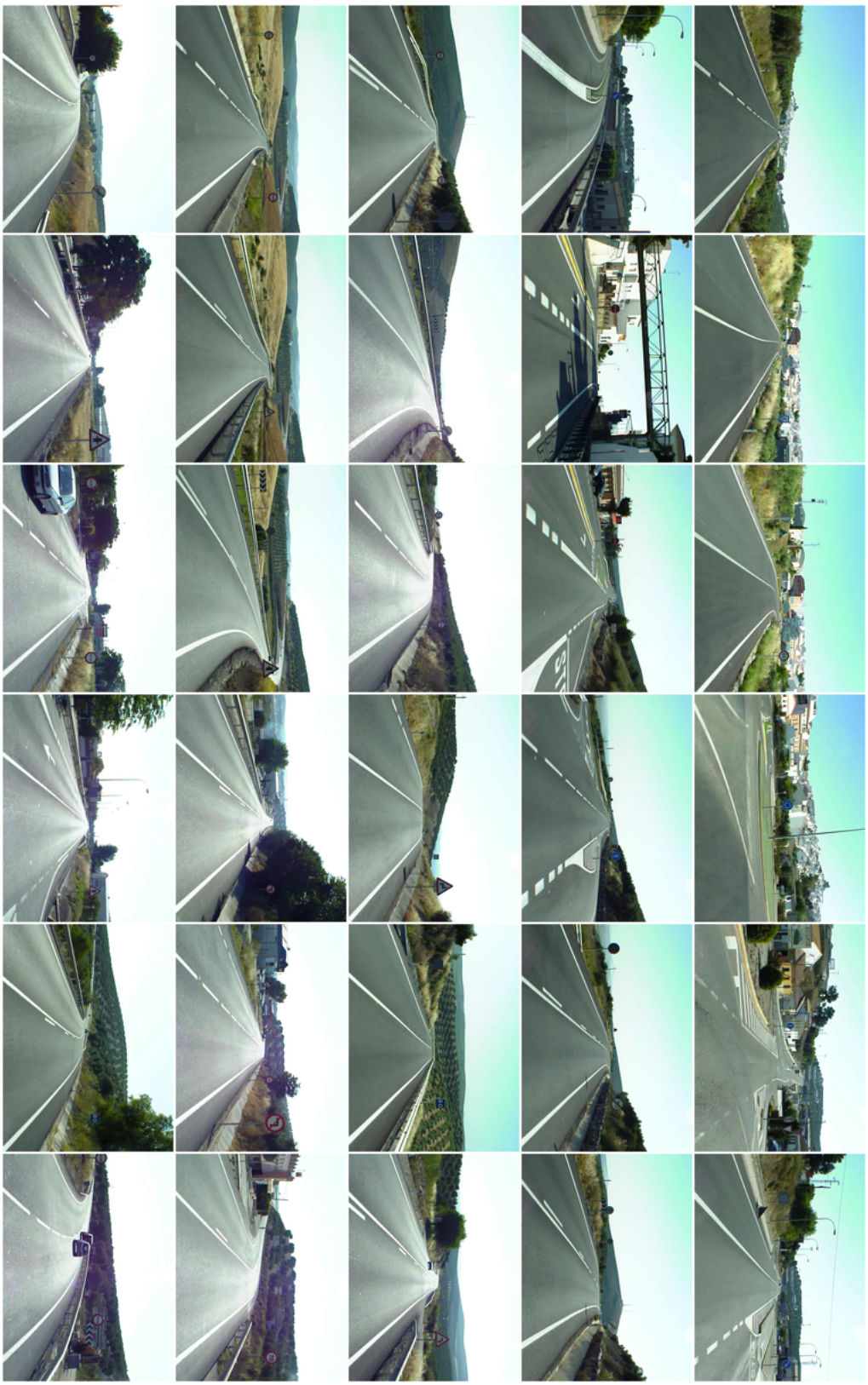


Figura 4.28: Algunas imágenes correspondientes a la secuencia para las pruebas de tracking.

- **Número de señales perdidas**. El número de señales que no se han reconocido de ninguna manera para un determinado método de segmentación.
- **Número de máximos**. Este parámetro indica el número de veces que un método ha alcanzado el máximo en un tipo de señal.
- **Porcentaje de falsos reconocimientos**. Esta cifra representa el porcentaje de señales erróneamente reconocidas por un método con respecto al total de señales reconocidas.
- **Velocidad**. Mide el tiempo de ejecución por imagen. Es decir, el tiempo total de ejecución (en segundos) se divide por el número de imágenes usadas.

Estas medidas se realizarán mediante el uso de una herramienta automática que comparará los resultados obtenidos con cada método de segmentación con los datos obtenidos de los conjuntos mediante observación. Esta herramienta automática usa Matlab y comandos "bash" en Linux para extraer y procesar los datos. Todas las medidas se han realizado con el kernel 2.6.27 en una distribución Ubuntu 8.10.

Los resultados se presentarán en tablas donde cada columna representa un método de segmentación y cada fila un determinado tipo de señal. El número de fila identifica el tipo de señal según se ve en la figura 4.25. Los mejores resultados para cada tipo de señal se presentan en negrita para una fácil identificación. Además de estas tablas se presentarán los datos de manera gráfica para facilitar la comparación de los métodos.

### 4.3.5. Resultados

**Métodos de descomposición acromática.** En primer lugar hay que comprobar si la descomposición Cromático/Acromática propuesta produce buenos resultados y cuales de los métodos presentados en ese apartado son los mejores. Lo haremos en primer lugar para decidir que método acromático se empareja con los cromáticos propuestos ya que posteriormente se utilizarán de manera conjunta.

Los resultados obtenidos aparecen en la tabla 4.10. En esta tabla sólo aparecen señales con algún contenido de blanco ya que la descomposición acromática por si sola únicamente permite segmentar ese color. Información adicional se muestra en la tabla 4.11.

En la tabla 4.11, se observa que el índice CAD original ([Liu02]) no es una opción para obtener los mejores resultados. El índice CAD modificado que separa las tres diferencias en RGB mejora los resultados pero aún así no son los mejores. El método que obtiene los mejores datos en Puntuación total, Porcentaje de reconocimiento, Número de señales perdidas y Número de máximos es el RGB normalizado. Sin embargo, las diferencias con el método basado en el espacio Ohta o el que utiliza la saturación y la intensidad (SI) son mínimas. En cuanto a falsos positivos no hay un método claramente mejor. En velocidad todos los métodos son muy similares aunque el CAD original alcanza los mejores datos.

Basándonos en estos datos, se decidió usar el método acromático RGB normalizado y el basado en el espacio Ohta en conjunción con sus contrapartidas de color, y el SI acromático con los métodos HST y HSET. Aunque se podría haber utilizado el RGB normalizado con el resto de métodos de color, esto hubiera implicado el uso de dos espacios de color a la vez y, puesto que los resultados de los métodos acromáticos elegidos no son muy diferentes, no se consideró como una opción válida.

Tabla 4.10: Resultados del número de señales reconocidas para los diferentes métodos acromáticos. Cada fila representa una señal con contenido en blanco dentro del conjunto de pruebas.

| **Señal** | RGBN | Ohta | SI | CAD | RGB |
|---|---|---|---|---|---|
| 1 | 0.72 | 0.44 | ***0.78*** | 0.00 | 0.67 |
| 4 | ***0.67*** | ***0.67*** | 0.44 | 0.44 | 0.00 |
| 5 | ***0.14*** | ***0.14*** | ***0.14*** | ***0.14*** | ***0.14*** |
| 6 | ***1.00*** | 0.93 | 0.93 | 0.13 | 0.60 |
| 8 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 9 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 10 | ***0.88*** | 0.75 | 0.75 | 0.62 | 0.75 |
| 11 | 0.62 | 0.62 | 0.50 | 0.38 | ***0.75*** |
| 12 | ***0.62*** | ***0.62*** | ***0.62*** | 0.25 | ***0.62*** |
| 13 | ***0.45*** | ***0.45*** | ***0.45*** | 0.36 | 0.36 |
| 16 | 0.80 | ***0.90*** | ***0.90*** | 0.80 | ***0.90*** |
| 17 | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | 0.57 |
| 18 | 0.36 | ***0.71*** | 0.57 | 0.43 | 0.36 |
| 20 | 0.75 | 0.62 | 0.62 | 0.75 | ***1.00*** |
| 21 | ***1.00*** | 0.62 | 0.75 | 0.88 | 0.62 |
| 22 | ***0.88*** | 0.81 | 0.81 | 0.69 | ***0.88*** |
| 25 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 26 | 0.82 | ***0.89*** | 0.82 | 0.61 | 0.41 |
| 28 | ***0.90*** | ***0.90*** | ***0.90*** | ***0.90*** | ***0.90*** |
| 29 | ***0.68*** | ***0.68*** | ***0.68*** | ***0.68*** | ***0.68*** |

Tabla 4.11: Análisis de los resultados de la tabla 4.10.

| **Medidas** | RGBNT | Ohta | SI | CAD | RGB |
|---|---|---|---|---|---|
| Puntuación total | ***12.29*** | 11.78 | 11.69 | 9.07 | 10.22 |
| Reconocimiento ( %) | ***65.66*** | 64.53 | 64.15 | 47.55 | 51.32 |
| Perdidas | ***3*** | ***3*** | ***3*** | 4 | 4 |
| Máximos | ***11*** | 10 | 8 | 4 | 8 |
| Falsos ( %) | ***0.00*** | ***0.00*** | ***0.00*** | 1.56 | 2.86 |
| Velocidad (seg/imagen) | 0.1478 | 0.1356 | 0.1400 | ***0.1130*** | 0.1340 |

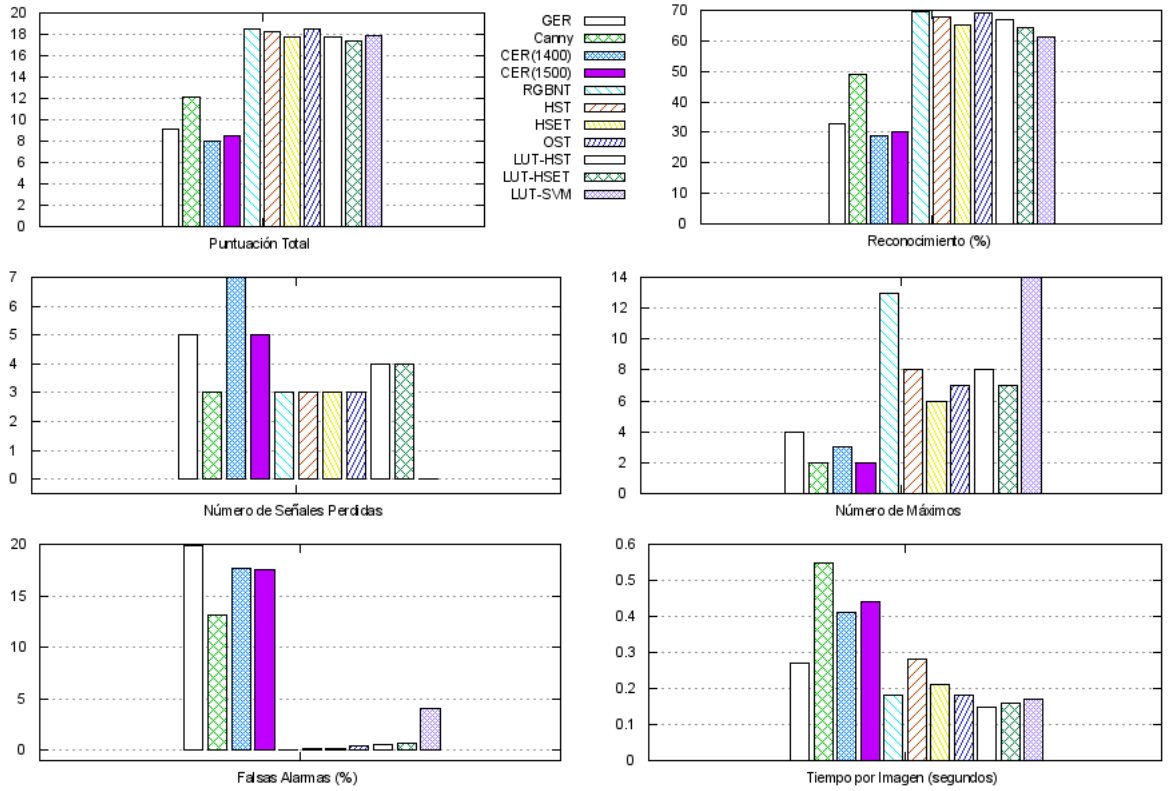



Figura 4.29: Resultados gráficos de los datos presentados en la tabla 4.13.

**Métodos de segmentación de color.** En este apartado se presentarán los resultados obtenidos mediante los métodos cromáticos combinados con los acromáticos que se han seleccionado anteriormente. En este caso los resultados corresponden a todas las posibles señales dentro del conjunto de pruebas incluyendo las que contienen colores rojo, blanco, azul y amarillo. La presentación se realiza mediante tablas con los datos en bruto (Tabla 4.12), tablas de datos elaborados a partir de los anteriores (Tabla 4.13) y gráficas que permiten una comparación más sencilla (Figura 4.29).

De todos esos datos se puede comprobar que los mejores métodos de segmentación son los de Umbralizado de Espacios de Color (CST). Los métodos de detección de bordes no son los mejores en todos los casos, pero para algunas señales, como la 5 que es sólo blanca (Fin de prohibición) y es problemática en la descomposición acromática, pueden ser una opción. De estos últimos métodos, el de Canny es el que mejor funciona.

Si analizamos los métodos CST, el RGB normalizado obtiene el mejor resultado aunque el resto de métodos obtiene resultados similares y, por ello, no se pueden descartar totalmente. Los mejores resultados en señales perdidas y número de máximos los obtiene la LUT SVM, es decir, el método de segmentación mediante SVM acelerado mediante una tabla de búsqueda. El porcentaje de falsas detecciones es muy similar entre los métodos de color aunque es mayor para el caso de la LUT SVM. En los métodos de detección por bordes este último parámetro es excesivo.

En velocidad de ejecución los mejores datos se obtienen con los métodos CST mientras que los métodos de detección de bordes son significativamente peores. Esto es debido a que el número de posibles señales candidatas es mayor en estos métodos. Dentro de los métodos

Tabla 4.12: Resultados del número de señales reconocidas para los distintos métodos estudiados. Son resultados normalizados respecto al máximo posible. Cada fila representa a una posible señal.

| **Señal** | GER | Canny | CER (1400) | CER (1500) | RGBNT | HST | HSET | OST | LUT HST | LUT HSET | LUT SVM |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.03 | 0.44 | 0.36 | 0.36 | 0.50 | 0.58 | 0.58 | 0.53 | 0.42 | 0.39 | ***0.64*** |
| 2 | 0.47 | 0.53 | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** |
| 3 | 0.17 | 0.94 | 0.36 | 0.39 | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** |
| 4 | 0.06 | 0.33 | 0.39 | 0.33 | 0.11 | 0.39 | ***0.44*** | ***0.44*** | 0.33 | 0.39 | 0.17 |
| 5 | ***0.71*** | 0.29 | 0.43 | 0.57 | 0.14 | 0.14 | 0.14 | 0.14 | 0.00 | 0.00 | 0.14 |
| 6 | 0.50 | ***1.00*** | 0.50 | 0.50 | 0.97 | ***1.00*** | 0.93 | 0.93 | 0.97 | 0.97 | 0.93 |
| 7 | 0.19 | 0.63 | 0.15 | 0.22 | ***0.70*** | 0.67 | 0.63 | 0.67 | 0.67 | 0.63 | 0.59 |
| 8 | 0.09 | 0.16 | 0.03 | 0.03 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ***0.47*** |
| 9 | 0.11 | 0.22 | ***0.33*** | 0.22 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.22 |
| 10 | 0.19 | 0.31 | 0.19 | 0.38 | ***0.81*** | 0.69 | 0.69 | 0.75 | 0.75 | 0.75 | 0.69 |
| 11 | 0.06 | 0.19 | 0.06 | 0.06 | 0.31 | 0.38 | 0.38 | 0.38 | 0.31 | 0.31 | ***0.50*** |
| 12 | 0.31 | 0.31 | 0.31 | 0.31 | ***0.62*** | 0.56 | 0.56 | 0.50 | ***0.62*** | 0.56 | 0.56 |
| 13 | 0.27 | 0.32 | 0.27 | 0.23 | 0.55 | 0.45 | 0.45 | 0.55 | 0.45 | 0.41 | ***0.59*** |
| 14 | ***1.00*** | 0.35 | 0.25 | 0.25 | 0.53 | 0.40 | 0.25 | 0.50 | 0.42 | 0.20 | 0.57 |
| 15 | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** |
| 16 | 0.60 | 0.15 | 0.00 | 0.00 | ***1.00*** | 0.85 | 0.90 | 0.90 | 0.90 | 0.85 | 0.70 |
| 17 | 0.36 | 0.39 | 0.43 | 0.43 | 0.82 | 0.86 | 0.79 | 0.86 | ***0.89*** | 0.86 | 0.39 |
| 18 | ***0.89*** | 0.39 | 0.00 | 0.07 | ***0.89*** | 0.75 | 0.71 | 0.82 | 0.75 | 0.75 | 0.57 |
| 19 | 0.00 | 0.00 | 0.00 | 0.00 | 0.18 | 0.09 | 0.18 | 0.18 | 0.09 | ***0.27*** | 0.09 |
| 20 | 0.00 | 0.25 | 0.06 | 0.06 | 0.50 | ***0.62*** | 0.50 | 0.56 | 0.44 | 0.44 | ***0.62*** |
| 21 | 0.50 | 0.50 | 0.38 | 0.50 | 0.69 | 0.75 | 0.69 | 0.75 | 0.56 | 0.56 | ***0.88*** |
| 22 | 0.12 | 0.38 | 0.19 | 0.25 | ***0.66*** | ***0.66*** | 0.62 | 0.59 | 0.62 | 0.62 | 0.34 |
| 23 | 0.17 | 0.87 | 0.30 | 0.30 | ***1.00*** | ***1.00*** | 0.87 | ***1.00*** | 0.96 | 0.87 | ***1.00*** |
| 24 | 0.00 | 0.00 | 0.00 | 0.00 | 0.95 | 0.95 | ***1.00*** | 0.95 | ***1.00*** | ***1.00*** | ***1.00*** |
| 25 | 0.00 | 0.00 | 0.00 | 0.07 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ***0.29*** |
| 26 | 0.28 | 0.65 | 0.45 | 0.45 | ***0.86*** | 0.83 | 0.82 | ***0.86*** | 0.84 | 0.84 | 0.42 |
| 27 | 0.93 | 0.71 | 0.50 | 0.50 | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** | ***1.00*** |
| 28 | 0.05 | 0.50 | 0.00 | 0.00 | 0.95 | 0.95 | 0.95 | 0.95 | ***1.00*** | ***1.00*** | ***1.00*** |
| 29 | 0.00 | 0.36 | 0.00 | 0.00 | ***0.76*** | 0.70 | 0.64 | 0.70 | 0.74 | 0.66 | 0.48 |

Tabla 4.13: Análisis de los resultados de la tabla 4.12.

| **Medidas** | GER | Canny | CER (1400) | CER (1500) | RGBNT | HST | HSET | OST | LUT HST | LUT HSET | LUT SVM |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Conjunto completo incluyendo información de color y de blanco** | | | | | | | | | | | |
| Puntuación total | 9.07 | 12.18 | 7.95 | 8.50 | 18.51 | 18.27 | 17.74 | ***18.52*** | 17.75 | 17.33 | 17.87 |
| Reconocimiento (%) | 33.01 | 48.82 | 28.99 | 30.10 | ***69.49*** | 67.96 | 65.33 | 68.93 | 66.99 | 64.49 | 61.44 |
| Perdidas | 5.00 | 3.00 | ***7.00*** | 5.00 | 3.00 | 3.00 | 3.00 | 3.00 | 4.00 | 4.00 | 0.00 |
| Máximos | 4.00 | 2.00 | 3.00 | 2.00 | 13.00 | 8.00 | 6.00 | 7.00 | 8.00 | 7.00 | ***14.00*** |
| Falsos (%) | 19.87 | 13.09 | 17.72 | 17.49 | 0.00 | 0.20 | 0.21 | 0.40 | 0.62 | 0.64 | 4.11 |
| Velocidad (seg/imagen) | 0.27 | 0.55 | 0.41 | 0.44 | 0.18 | 0.28 | 0.21 | 0.18 | **0.15** | 0.16 | 0.17 |
| **Sólo información de blanco** | | | | | | | | | | | |
| Puntuación total | 4.56 | 8.24 | 5.64 | 6.15 | ***10.36*** | 9.43 | 9.43 | 10.17 | 9.22 | 9.22 | 9.00 |
| Reconocimiento (%) | 32.17 | 56.52 | 41.30 | 42.17 | ***62.17*** | 59.13 | 59.13 | 60.00 | 56.96 | 56.96 | 42.17 |
| Perdidas | 6.00 | 3.00 | 5.00 | 4.00 | 3.00 | 4.00 | 4.00 | 4.00 | 5.00 | 5.00 | ***2.00*** |
| Máximos | 3.00 | 7.00 | 2.00 | 2.00 | ***10.00*** | 5.00 | 5.00 | 9.00 | 4.00 | 4.00 | 9.00 |
| Falsos (%) | 13.95 | 1.52 | 3.06 | 5.83 | 1.38 | 1.45 | 1.45 | 1.43 | ***0.76*** | ***0.76*** | 14.16 |
| **Sólo información de color** | | | | | | | | | | | |
| Puntuación total | 5.48 | 7.53 | 5.21 | 5.23 | 16.42 | ***16.62*** | 15.99 | 16.03 | ***16.62*** | 15.95 | 15.92 |
| Reconocimiento (%) | 28.98 | 41.57 | 24.94 | 25.18 | ***71.50*** | 70.78 | 68.17 | 70.31 | 70.78 | 67.70 | 70.55 |
| Perdidas | 9.00 | 12.00 | 12.00 | 12.00 | 3.00 | 3.00 | 3.00 | 4.00 | 3.00 | ***2.00*** | ***2.00*** |
| Máximos | 3.00 | 1.00 | 1.00 | 1.00 | ***11.00*** | 9.00 | 10.00 | 10.00 | 8.00 | 10.00 | 10.00 |
| Falsos (%) | 22.29 | 19.72 | 24.46 | 22.63 | 1.31 | ***0.00*** | ***0.00*** | 1.00 | 0.67 | 0.35 | 1.00 |

CST, el HST tiene la peor velocidad aunque se puede mejorar mediante el uso de tablas de búsqueda. El método RGB normalizado y el del espacio Ohta (OST) son bastante rápidos aunque no usan tabla de búsqueda. Las tablas de búsqueda obtienen buenos resultados de velocidad y, las diferencias que hay entre ellas son debidas al distinto número de objetos detectados según el método. Si ese número es alto la detección y el reconocimiento tardan más en seleccionar.

La mejora de velocidad mediante tablas presentada en la página 122 se demuestra que es efectiva en ese parámetro pero además no reduce la calidad significativamente. Esto puede verse al comparar los métodos HST y LUT HST o HSET y LUT HSET.

**Ajuste de umbrales. Sensibilidad.** Los resultados presentados hasta ahora dependen claramente de los valores a los que se han ajustado los umbrales. Aunque los experimentos se han realizado buscando los mejores resultados, es posible que otros mejores se puedan obtener variando dichos umbrales. Por ello, se ha diseñado un ajuste más complejo para aquellos métodos que dependen fuertemente de umbrales, como los acromáticos o los CST.

El método diseñado consta de dos pasos:

- **Ajuste empírico inicial.** El primer ajuste de un determinado método de segmentación se inicia buscando los umbrales de una manera gráfica, es decir, cuando los umbrales referidos al rojo se ajustan, se elige una imagen con una señal roja y se tocan los umbrales hasta que se obtiene una buena segmentación de manera visual, de igual manera se haría para el resto de colores. Después de que todos los colores se han ajustado de esta manera se obtienen resultados de reconocimiento para un determinado conjunto. Posteriormente, los umbrales se refinan para mejorar esos resultados en un procedimiento de prueba y error.

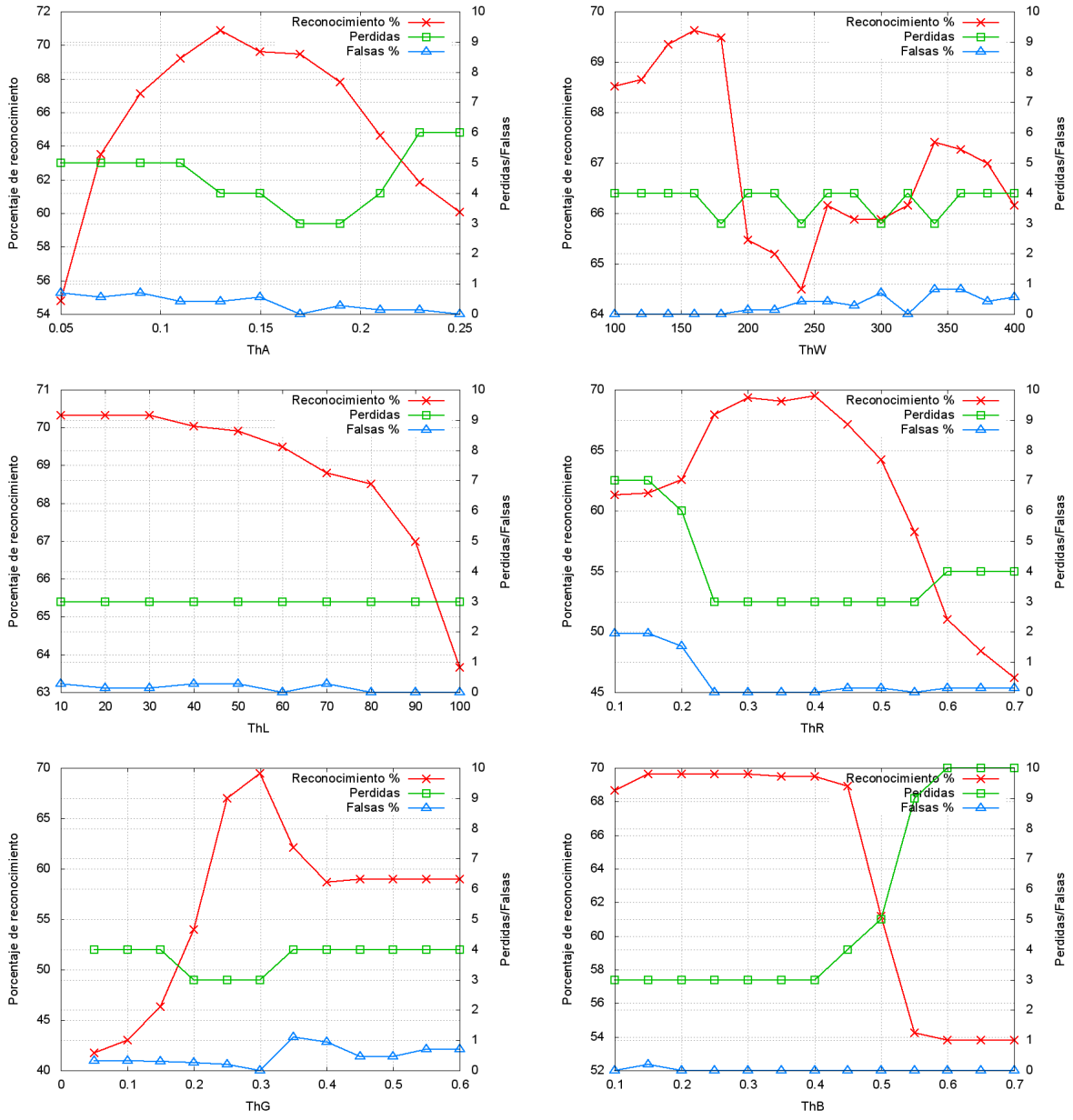



Figura 4.30: Variación de los resultados de reconocimiento. Ejemplos para RGB normalizado y umbrales $ThA$, $ThW$, $ThL$, $ThR$, $ThG$ y $ThB$. Las gráficas tienen dos ejes de ordenadas con diferentes escalas. La izquierda es para el porcentaje de reconocimiento y la derecha es para el porcentaje de perdidas y falsas.

- **Búsqueda exhaustiva.** En este segundo paso se realiza un barrido alrededor de los umbrales empíricos. En este barrido se obtienen resultados de reconocimiento para distintos valores de un umbral mientras el resto se mantienen fijos. De esta manera, se realiza una evaluación de reconocimiento para cada umbral. Representando esos valores en una gráfica se puede encontrar, mediante inspección, el mejor valor de ajuste.

La figura 4.30 muestra algunos ejemplos de las gráficas obtenidas de esta manera. Estos ejemplos se corresponden con el método en RGB normalizado y representan la evolución del porcentaje de reconocimiento, las señales perdidas y el porcentaje de falsas detecciones con respecto a los umbrales $ThA$, $ThW$, $ThL$, $ThR$, $ThG$ y $ThB$. Se representan estos tres parámetros en una única gráfica puesto que es necesario llegar a un compromiso entre la detección de falsos y correctos y las señales perdidas.

Mirando, por ejemplo, a la gráfica de $ThA$ se puede ver que $ThA = 0.17$ da buenos resultados de reconocimiento, sólo se pierden 3 señales y no hay falsas detecciones. Sin embargo, los mejores resultados de reconocimiento se dan para $ThA = 0.13$ pero con un porcentaje de falsas detecciones y una señal más perdida. En este caso el ajuste empírico inicial dio un valor de $ThA = 0.14$ que es próximo al óptimo pero no es el mejor. Por ello, este umbral se modificó al valor óptimo. La inspección de cada gráfica para los diferentes umbrales y métodos da una información similar a la del ejemplo y, aunque algunas veces el valor de ajuste empírico se modificó la mayoría de las veces era el ideal.

Estas gráficas además permiten apreciar la sensibilidad de cada método de segmentación respecto a los umbrales de los que dependen. Esto puede ser interesante como un parámetro más de calidad que permita discriminar entre métodos de segmentación con resultados parecidos.

**Validación.** Los datos obtenidos hasta este punto son bastante significativos para obtener algunas conclusiones sobre qué métodos de segmentación deberían usarse en la tarea del reconocimiento de señales de tráfico. Sin embargo, puesto que los conjuntos de señales de prueba inicial se han usado tanto para obtener resultados como para ajustar parámetros, queda la duda de si dichos resultados podrían generalizarse a otras imágenes distintas. Por ello se creó un nuevo conjunto de validación (página 123).

Los datos de calidad obtenidos para ese nuevo conjunto se muestran en forma de tabla (Tabla 4.14) y de forma gráfica para facilitar la comparación (Figura 4.31). Si se comparan estos nuevos datos con los ya obtenidos se aprecia una confirmación básica de lo apreciado. El método RGB normalizado sigue siendo el mejor en la mayoría de parámetros y los resultados de los métodos CST y los de detección de bordes son similares. La única diferencia importante aparece en el método SVM ya que los resultados de validación son peores (Las SVM no se debían entrenar de nuevo para esta validación). Un análisis más profundo de los datos separados en color y blanco muestra que hay una reducción en el porcentaje de reconocimiento de las señales con contenido blanco mientras que las de color se mantienen. Esta reducción puede ser debida a un mal entrenamiento en el caso del color blanco ya que el entrenamiento de color muestra una buena generalización mientras que el blanco no. Esto puede considerarse como un inconveniente del método SVM ya que para obtener buenos resultados para el blanco es necesario un entrenamiento más elaborado

Tabla 4.14: Resultados de los conjuntos de validación.

| **Medidas** | GER | Canny | CER (1400) | CER (1500) | RGBNT | HST | HSET | OST | LUT HST | LUT HSET | LUT SVM |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Conjunto completo incluyendo información de color y de blanco** | | | | | | | | | | | |
| Puntuación total | 17.47 | 23.78 | 19.27 | 17.85 | ***34.26*** | 32.24 | 31.40 | 33.37 | 33.13 | 32.17 | 27.05 |
| Reconocimiento ( %) | 44.08 | 59.87 | 45.07 | 43.75 | 77.96 | 75.33 | 73.68 | ***78.29*** | 75.99 | 73.68 | 58.22 |
| Perdidas | 15.00 | 10.00 | 12.00 | 13.00 | ***3.00*** | ***3.00*** | ***3.00*** | 4.00 | ***3.00*** | ***3.00*** | 5.00 |
| Máximos | 10.00 | 16.00 | 9.00 | 8.00 | ***28.00*** | 25.00 | 23.00 | 26.00 | ***28.00*** | 25.00 | 16.00 |
| Falsos ( %) | 16.25 | 26.61 | 24.73 | 22.67 | 1.66 | 2.14 | 2.61 | 3.25 | 1.28 | ***0.88*** | 8.76 |
| Velocidad (seg/imagen) | 0.3 | 0.69 | 0.52 | 0.4 | 0.18 | 0.31 | 0.21 | 0.16 | **0.14** | **0.14** | 0.15 |
| **Sólo información de blanco** | | | | | | | | | | | |
| Puntuación total | 11.08 | 16.32 | 10.80 | 10.14 | 18.18 | ***18.98*** | ***18.98*** | 18.35 | 18.91 | 18.91 | 4.97 |
| Reconocimiento ( %) | 43.38 | 61.76 | 41.91 | 41.91 | 70.59 | 70.59 | 70.59 | ***71.32*** | 70.59 | 70.59 | 23.53 |
| Perdidas | 9.00 | ***5.00*** | 10.00 | 10.00 | 6.00 | ***5.00*** | ***5.00*** | 6.00 | ***5.00*** | ***5.00*** | 18.00 |
| Máximos | 9.00 | 12.00 | 5.00 | 4.00 | 15.00 | ***16.00*** | ***16.00*** | ***16.00*** | ***16.00*** | ***16.00*** | 2.00 |
| Falsos ( %) | 6.35 | 23.64 | 25.97 | 25.00 | 4.95 | 4.00 | 4.00 | 3.96 | ***3.03*** | ***3.03*** | 17.95 |
| **Sólo información de color** | | | | | | | | | | | |
| Puntuación total | 13.91 | 20.51 | 19.84 | 18.74 | 33.39 | 32.89 | 31.15 | ***33.76*** | 32.95 | 30.92 | 32.84 |
| Reconocimiento ( %) | 44.05 | 58.33 | 46.43 | 44.64 | ***85.12*** | 80.95 | 76.19 | 83.93 | 79.76 | 75.60 | 82.74 |
| Perdidas | 19.00 | 11.00 | 12.00 | 14.00 | ***4.00*** | ***4.00*** | ***4.00*** | ***4.00*** | ***4.00*** | ***4.00*** | 5.00 |
| Máximos | 8.00 | 13.00 | 11.00 | 11.00 | ***32.00*** | 29.00 | 24.00 | 31.00 | 29.00 | 24.00 | 29.00 |
| Falsos ( %) | 21.28 | 26.32 | 22.77 | 20.21 | ***0.69*** | 0.73 | 1.54 | 0.70 | 1.47 | 0.78 | 2.80 |

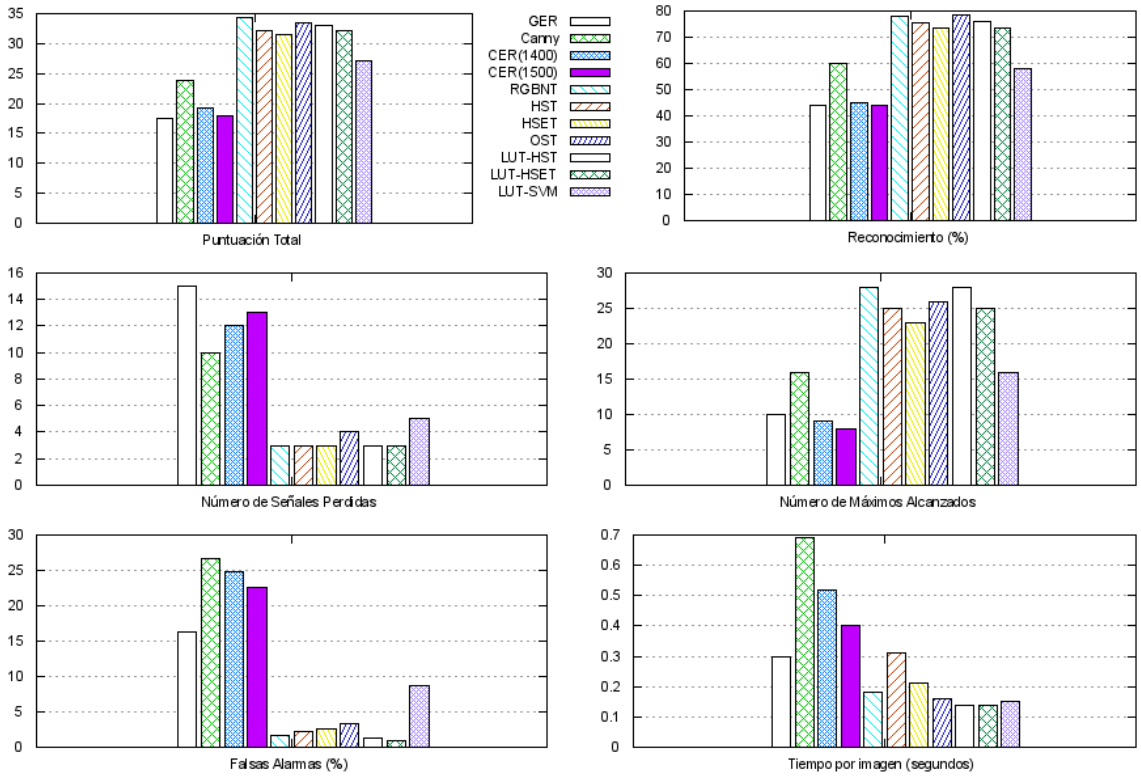



Figura 4.31: Resultados gráficos para el conjunto de validación en la tabla 4.14.

Tabla 4.15: Resultados usando la información de tracking

| **Medidas** | GER | Canny | CER (1400) | CER (1500) | RGBNT | HST | HSET | OST | LUT HST | LUT HSET | LUT SVM |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Total | ***47*** | 20 | 21 | 23 | 46 | 45 | 45 | ***47*** | 42 | 42 | 42 |
| Correctas | 41 | 17 | 18 | 18 | 46 | 44 | 45 | ***47*** | 42 | 42 | 42 |
| Falsas | 6 | 3 | 3 | 5 | ***0*** | 1 | ***0*** | ***0*** | ***0*** | ***0*** | ***0*** |
| Perdidas | 6 | 30 | 29 | 29 | 1 | 3 | 2 | ***0*** | 5 | 5 | 5 |

que para color. Posiblemente sea necesario realizar una clasificación de píxeles cromáticos y acromáticos como en el resto de métodos.

**Resultados de Tracking.** Finalmente sólo queda probar el sistema completo, incluyendo el tracking, variando el sistema de segmentación. Para ello se utilizará la secuencia presentada en el apartado 4.3.3.

En esta prueba no es posible realizar una recopilación exhaustiva de datos como en las pruebas precedentes, ya que el número de imágenes es mucho más alto. Por eso, en este caso, se toma como salida las señales que han sido detectadas por el sistema de tracking, que, como se dijo, agrupa las sucesivas detecciones de una misma señal. En la tabla 4.15 se muestran los datos obtenidos en este caso.

La primera fila muestra el número de señales realmente detectadas y que han entrado en el tracking, la segunda indica las señales que fueron correctamente identificadas, la tercera el número de falsas señales y la última muestra el número de señales perdidas respecto al método que da el mayor número de señales correctas.

De estos resultados se puede deducir que el método OST es el mejor ya que no da señales perdidas ni falsas. Sin embargo, el RGB normalizado obtiene unos resultados muy similares, con sólo una señal perdida. Los métodos HST y HSET dan buenos resultados aunque tienen 2 señales perdidas y una y tres falsas respectivamente. Las tablas de búsqueda obtienen los peores resultados perdiendo 5 señales.

Una vez más, los métodos CST se comportan mejor que los de detección de bordes y aunque el método GER tiene, comparativamente, buenos resultados también produce demasiadas señales perdidas o falsas alarmas. El método Canny, que obtenía buenos resultados (entre los métodos de detección de bordes) en las pruebas anteriores aquí se comporta peor. Una posible explicación puede ser que este método depende fuertemente del ajuste de sus parámetros.

### 4.3.6. Discusión

Del análisis de los datos obtenidos podemos destacar lo siguiente:

1. El porcentaje de reconocimiento para el mejor método es del 69.49 % para el conjunto de prueba y del 78.29 % para el conjunto de validación. Estos resultados pueden parecer bajos pero han sido obtenidos teniendo en cuenta todas las veces que una señal puede ser detectada dentro de una secuencia (en cada imagen y para todos

los colores posibles). Además, las imágenes han sido seleccionadas por su complejidad y sus problemas respecto a la segmentación y, por tanto, el porcentaje de reconocimiento tiende a ser bajo.

2. Para las pruebas realizadas con el conjunto de pruebas y el de validación el mejor método es el RGB Normalizado mientras que para el conjunto de Tracking los mejores datos se obtienen para el método OST. La Tabla que usa las SVM es una opción válida puesto que obtiene un 82.54 % con sólo la información de color, aunque podría mejorar con un entrenamiento mejor para el caso del blanco.

3. Los métodos de detección de bordes se pueden usar como complemento a otro método de color pero no de manera aislada.

4. El uso de tablas de búsqueda reducidas para mejorar la velocidad es efectivo puesto que la calidad es similar al método original.

5. Ningún método se comporta bien en todos los contextos.

6. La normalización como en RGB Normalizado o OST mejora los resultados y es una operación muy sencilla. Aunque HST y HSET tienen buenos resultados, su costo en velocidad les hace innecesarios. ¿Porqué usar una transformación compleja si una simple normalización es suficiente?

Los resultados presentados en los apartados anteriores deben entenderse como útiles para comparar métodos y, en ningún caso, deben tomarse como absolutos e inamovibles.

## 4.4. Resumen

En este capítulo se ha presentado un método de segmentación en color basado en la clasificación con SVM. Para comprobar su efectividad se ha comparado con otros métodos en la tarea de segmentación dentro de un sistema de clasificación de señales de tráfico. Los resultados obtenidos por dicho método han sido buenos o incluso mejores que los obtenidos por otros métodos. Sin embargo, queda pendiente la mejora de la clasificación de los colores acromáticos o bien mediante un nuevo entrenamiento más exhaustivo o implementando la descomposición Cromático/Acromático mediante las SVM.

El uso de una tabla de búsqueda reducida para poder aplicar las SVM en tiempo real también puede influir en sus resultados aunque de forma reducida. Una reducción de vectores soporte usando nuevas técnicas de entrenamiento podría suponer el uso directo de las SVM sin la tabla de búsqueda con cierta mejora en sus resultados.

# Capítulo 5

# Eliminación de ruido impulsivo

La eliminación de ruido es una de las más importantes áreas dentro del procesado digital de imágenes y un problema clásico en este campo. El ruido impulsivo es uno de los tipos de ruido que pueden aparecer en las imágenes. Este ruido se puede generar en la imagen en su adquisición, su almacenamiento o su transmisión y puede afectar a etapas posteriores de procesado si no se elimina preservando los detalles ([Mélange11, Akkoul10, Xu09, Abreu96]). Este problema se puede presentar en diversos campos, desde las imágenes médicas [Toprak07] al análisis de imágenes de satélite [Alamri10].

Se han propuesto un gran número de técnicas para eliminar o reducir este tipo de ruido. Del análisis de esas técnicas se deduce, que el esquema basado en una primera detección de los píxeles que son ruidosos y una posterior sustitución de estos es el que produce mejores resultados. En este capítulo de la tesis se demuestra que el uso de clasificadores de tipo neuronal como Funciones de Base Radial (Radial Basis Functions o RBF) o Máquinas de Vectores Soporte (SVM) para la detección del ruido produce buenos resultados. También se introduce el uso de SVM como función de regresión para la obtención de los valores de reconstrucción de los píxeles.

## 5.1. Estado del arte

En este apartado se presentan algunas de las técnicas de eliminación de ruido impulsivo más importantes clasificadas de acuerdo a la forma en que abordan el problema del filtrado. Sin embargo, una primera sección se dedica a la presentación del problema a tratar, así como, a fijar la notación empleada.

### 5.1.1. Definición del problema

Como se ha comentado previamente, uno de los tipos de ruido que pueden aparecer en las imágenes es el impulsivo debido a problemas en la adquisición, almacenamiento o transmisión de las mismas. Este tipo de ruido se caracteriza porque algunos píxeles de la imagen son reemplazados por valores anómalos mientras que el resto permanecen inalterados. Los valores anómalos pueden ser fijos, máximo o mínimo de la escala de grises, o pueden variar dentro de esa escala. El primer tipo es el conocido como "Salt&Pepper" y es el que se tratará en esta tesis.

Sea $I_{x,y}$ el valor de gris de la imagen original $I$ en un píxel localizado en la posición $(x, y)$ y $[n_{min}, n_{max}]$ el rango de la escala de grises de la imagen. Si llamamos $X$ a la imagen ruidosa ésta se caracterizará si presenta ruido impulsivo por la siguiente expresión:

$$X(x,y) = \begin{cases} I(x,y) & \text{con probabilidad } 1-p \\ R(x,y) & \text{con probabilidad } p \end{cases}, \tag{5.1}$$

donde, $R(x, y)$ es el valor de ruido que sustituye al píxel original en la posición $(x, y)$. Cuando el valor de sustitución es $R(x, y) \in \{[n_{min}, n_{max}]\}$, se dice que la imagen está corrompida por ruido impulsivo de valor aleatorio y cuando el valor es $R(x, y) \in n_{min}, n_{max}$ se dice que el ruido tiene valores de los impulsos fijos y se le conoce como ruido "Salt&Pepper". En el caso de imágenes en escala de grises con 8 bits por píxel los valores de sustitución para el ruido "Salt&Pepper" serían $n_{min}(0)$ o $n_{max}(255)$ mientras que en el otro tipo de ruido los valores de sustitución estarían en un rango uniforme entre 0 y 255.

### 5.1.2. Revisión de algoritmos

Como se ha comentado previamente, la eliminación del ruido impulsivo de las imágenes es un campo de investigación importante dentro del procesado digital de imágenes. En la web "scholar.google.es" la búsqueda de las palabras "salt and pepper" junto con "noise removal" da un total de 829 resultados, si a eso añadiéramos los artículos dedicados al ruido impulsivo en general el número sería muy superior.

El principal objetivo de todos los algoritmos propuestos es eliminar el ruido y mantener a la vez los detalles de la imagen. Podemos agrupar los distintos métodos en las dos siguientes categorías:

A. **Filtrado sin detección.** En este tipo de filtrado todos los píxeles de la imagen son restaurados independientemente de si son ruidosos o no. El esquema general de este tipo de filtrado es el uso de una máscara donde normalmente el centro es el píxel de interés. La máscara se mueve desde el extremo superior izquierdo al inferior derecho y realiza una serie de operaciones con los píxeles que hay dentro de la misma para obtener el valor de reconstrucción.

   A este grupo pertenece el filtrado de mediana que es la solución clásica para la eliminación de ruido impulsivo. Posteriormente han aparecido modificaciones de la mediana como el filtro "Center Weighted Median" (CWM) [Ko91, Yang95, Chen01] que repite el píxel central un número de veces especificado antes de realizar el ordenamiento. En [Astola97] se propone un tipo de filtro muy relacionado con los anteriores llamado AMF (Adaptative Median Filter) y que mejora las prestaciones del CWM. La principal ventaja de estos esquemas es su sencillez y su desventaja es la aparición de emborronado, sobre todo cuando el porcentaje de ruido es alto. Este efecto se produce porque el filtro cambia tanto los píxeles ruidosos como los no ruidosos, para evitar esto aparecen técnicas que primero detectan los píxeles ruidosos y que se comentan a continuación.

B. **Detección seguido de filtrado.** Este tipo de filtrado es el que se refleja en la figura 5.1. Cada píxel de la imagen pasa por un detector que decide si el píxel es ruidoso o no, y en caso de serlo un estimador cambia el valor original por el estimado. En caso

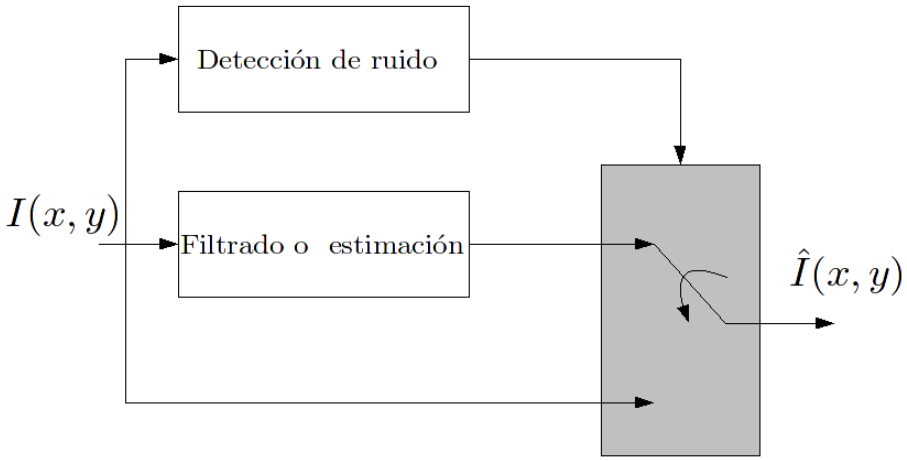



Figura 5.1: Esquema de filtrado de ruido impulsivo con detección

contrario el píxel no es modificado. La detección normalmente se realiza utilizando los píxeles alrededor del que se está tratando, y para ello se usa una máscara que se mueve por la imagen.

La mayoría de los algoritmos propuestos en la literatura se basan en este esquema, ya que el aplicar sólo la reconstrucción a los píxeles que son ruidosos mejora los resultados, reduciendo los efectos de emborronado que se puedan producir.

Los distintos métodos propuestos en esta categoría varían el tipo de detector y la forma en que el píxel ruidoso es sustituido.

Un buen número de ellos toman como base, tanto para la detección como para la recuperación, la mediana de los píxeles alrededor del píxel de interés. Por ejemplo, en [Chen99] se desarrolla un filtro no lineal llamado "Tri-state median" con el objetivo de eliminar el ruido pero manteniendo los detalles. Para ello se usa un filtro de mediana estándar y el CWM para la detección de píxeles ruidosos y se proponen tres estados posibles: píxel no modificado, píxel recuperado con la mediana y píxel recuperado con CWM. De esta manera se unen los mejores comportamientos de ambos filtros. Recientemente ha aparecido una mejora sobre este filtro en [Chang08].

Otro tipo de filtro similar a la mediana es el "Rank-Ordered Mean" (ROM) utilizado, por ejemplo, en [Lightstone95, Abreu96, Hasan00]. Se caracteriza por excluir el píxel de interés en los cálculos de la mediana.

Un importante grupo de métodos están basados en lógica difusa ([Zadeh65, Zadeh78]) tanto para la detección de los píxeles ruidosos como para su restauración. Los primeros trabajos se deben a F. Russo ([Russo96a, Russo96b, Russo95]) y los resultados obtenidos son buenos, con la ventaja de poderse aplicar a varios tipos de ruido incluido el gaussiano. Más recientemente en [Schulte07a, Schulte06b, Schulte07b, Schulte06a, Farbiz00] se presentan métodos de lógica difusa para la detección de ruido tanto en imágenes en blanco y negro como en color con mejores resultados que los anteriores.

Las redes neuronales también han sido una importante herramienta para la eliminación del ruido impulsivo [Yin91, Yin92, Sucher95a, Sucher94, Zhang01b] incluso mapas autoorganizados [Sucher95b] y mezcla de redes neuronales y lógica difusa [Russo00, Russo99]

**Filtro de mediana.** Como se ha comentado, la aparición de ruido y su eliminación han sido objeto de estudio desde los comienzos del procesado digital de imágenes. Cuando se trataba de ruido aditivo (gaussiano por ejemplo) una herramienta interesante era el filtrado de la imagen con filtros lineales paso bajo. Sin embargo, los filtros de suavizado lineales o filtros paso bajo tienden a "difuminar los ejes" ya que atenúan las altas frecuencias. La visión humana es muy sensible a esta información de alta frecuencia y, por tanto, la preservación y el posible realce de este detalle es muy importante al filtrar. Cuando el objetivo es más la reducción del ruido que el difuminado, el empleo de los filtros de mediana representan una posible alternativa ([González93, Pitas90]).

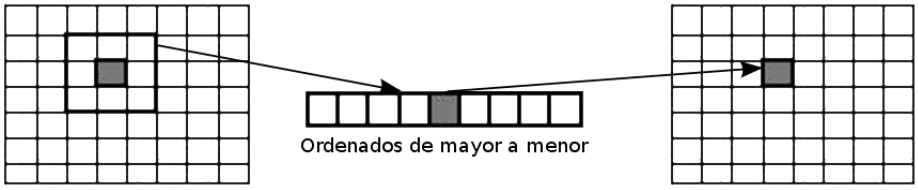


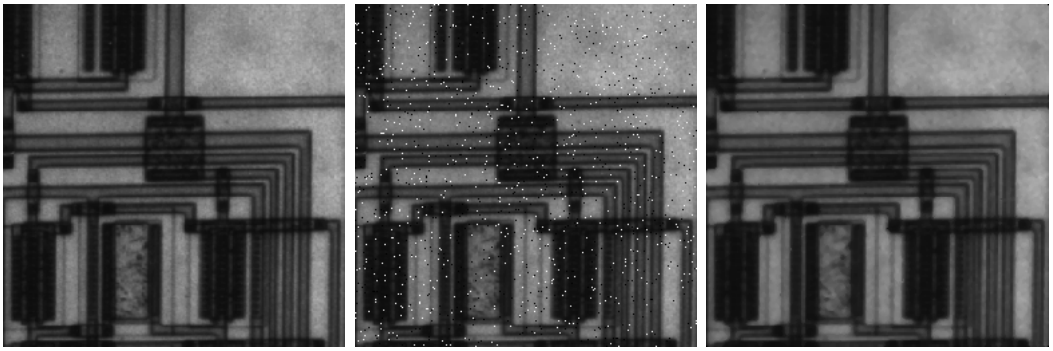

Figura 5.2: Esquema de filtrado de mediana $3 \times 3$

En el filtrado de mediana, el nivel de gris de cada píxel se reemplaza por la mediana de los niveles de gris en un entorno de este píxel, que se fija con un parámetro $N$ que nos dará el tamaño de la ventana alrededor del mismo, siendo este $N \times N$. En la figura 5.2 se puede ver un ejemplo gráfico en el caso de una ventana $3 \times 3$. En el caso de que el parámetro $N$ sea par, el valor que se toma es la media de los dos centrales.

Recordar que la mediana de un conjunto de valores es tal, que la mitad de los valores del conjunto son menores y la mitad de los valores mayores. Es decir en un conjunto ordenado de mayor a menor o viceversa, sería el valor de la posición central. El filtro de la mediana no puede ser calculado con una máscara de convolución, ya que es un filtro no lineal. Podemos ver cómo este tipo de filtro elimina los píxeles que tengan un valor muy diferente al resto de sus vecinos. Como se selecciona el valor central, el filtro de mediana es muy efectivo para eliminar píxeles cuyo valor es muy diferente del resto de sus vecinos, como en el caso del ruido impulsivo de la imagen.

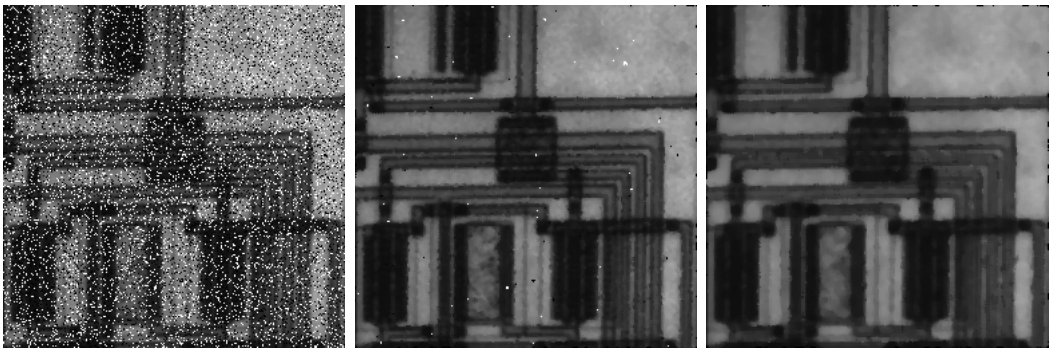
El principal problema de este tipo de filtrado es que se aplica por igual a píxeles ruidosos y no ruidosos. Esto no supone problema cuando el nivel de ruido es pequeño, ya que, con una ventana pequeña se puede eliminar el ruido manteniendo las altas frecuencias de la imagen. Sin embargo, si el ruido aumenta, es necesario aumentar el tamaño de la ventana para evitar que el pixel central sea ruidoso, eso hace que se produzca emborronado.

En la figura 5.3 tenemos un ejemplo de filtrado de mediana aplicando una ventana de $3 \times 3$ y con una cantidad de ruido reducida. Podemos comparar la imagen original (Figura 5.3(a)) con la recuperada con el filtrado (Figura 5.3(c)) comprobando que el ruido es eliminado y casi no se produce emborronado. Sin embargo, si el ruido aumenta (Figura



**(a)** **(b)** **(c)**

Figura 5.3: Ejemplo de filtrado de mediana con 2 % de ruido impulsivo. (a) Imagen original, (b) Imagen con 2 % de ruido, (c) Filtrado de Mediana con una ventana de $3 \times 3$



**(a)** **(b)** **(c)**

Figura 5.4: Ejemplo de filtrado de mediana con 20 % de ruido impulsivo. (a) Imagen con ruido, (b) Filtrado de Mediana con una ventana de $3 \times 3$, (c) Filtrado de Mediana con una ventana de $5 \times 5$

5.4(a)) la ventana de $3 \times 3$ es insuficiente para eliminar todo el ruido (Figura 5.4(b)). En la imagen de la figura 5.4(c) se amplia la ventana hasta una de $5 \times 5$, de forma que en este caso sí se elimina el ruido pero a costa de un mayor emborronamiento.

Aún con los problemas que presenta, la mediana es la base de un buen número de métodos tal y como se ha visto en la introducción de este apartado. Su principal utilidad está en la obtención del valor de reconstrucción cuando un píxel es considerado ruidoso.

**Filtros Rank Ordered Mean (ROM).** Entre las muchas variaciones de la mediana que se ha ido proponiendo en la literatura, destacan las que intentan mejorar sus resultados tratando de detectar primero los píxeles ruidosos y después cambiar sólo aquellos que lo sean (Esquema de detección seguido de filtrado).

Entre estos métodos, el propuesto en [Lightstone95, Abreu95] y luego ampliado en [Abreu96] se considera como un punto de referencia en el tema del filtrado de ruido impulsivo y es utilizado como comparación por muchos otros métodos.

Para la detección de pixeles ruidosos se utiliza el siguiente algoritmo:

1.- Se forma un vector $\mathbf{w}(\mathbf{n})$ con los píxeles pertenecientes a una ventana $3\times 3$ centrada en el píxel en cuestión $(x(n_1,n_2))$. El vector está formado por 8 elementos porque se excluye el pixel central.

$$\mathbf{w}(\mathbf{n}) = [w_1(\mathbf{n}), w_2(\mathbf{n}), ..., w_8(\mathbf{n})] = \begin{bmatrix} x(n_1-1,n_2-1) \\ x(n_1-1,n_2) \\ x(n_1-1,n_2+1) \\ x(n_1,n_2-1) \\ x(n_1,n_2+1) \\ x(n_1+1,n_2-1) \\ x(n_1+1,n_2) \\ x(n_1+1,n_2+1) \end{bmatrix}.$$

2.- Los elementos de $\mathbf{w}(\mathbf{n})$ se ordenan de menor a mayor formando un vector $\mathbf{r}(\mathbf{n}) = [r_1(\mathbf{n}), r_2(\mathbf{n}), \cdots, r_8(\mathbf{n})]$ de tal manera que se cumple $r_1(\mathbf{n}) \leq r_2(\mathbf{n}) \leq \cdots r_8(\mathbf{n})$.

3.- Ahora se puede definir la "Rank Ordered Mean" $(ROM)$ como $m(\mathbf{n}) = (r_4(\mathbf{n}) + r_5(\mathbf{n}))/2$. Como se puede apreciar la definición es muy parecida a la de la mediana pero excluyendo el píxel central.

4.- Se define un vector de diferencias llamado "rank-ordered differences" como $\mathbf{d}(\mathbf{n}) = [d_1(\mathbf{n}),\, d_2(\mathbf{n}), d_3(\mathbf{n}), d_4(\mathbf{n})]$ donde:

$$d_i(\mathbf{n}) = \begin{cases} r_i(\mathbf{n}) - x(\mathbf{n}) & x(\mathbf{n}) \leq m(\mathbf{n}) \\ x(\mathbf{n}) - r_{9-i}(\mathbf{n}) & x(\mathbf{n}) > m(\mathbf{n}) \end{cases},$$

con $i = 1, \cdots, 4$.

5.- Basándose en esas diferencias, el algoritmo detecta un píxel como ruidoso si se cumplen las siguientes desigualdades:

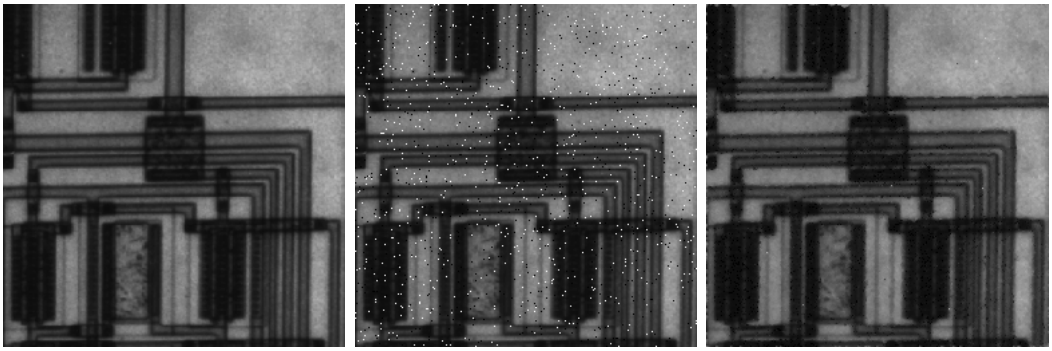
$$d_i(\mathbf{n}) > T_i,\ i = 1, \cdots, 4$$

donde $T_1, T_2, T_3, T_4$ son umbrales que cumplen $T_1 < T_2 < T_3 < T_4$. Si el pixel $x((n))$ es detectado como ruidoso se reemplaza por $m((n))$. Los umbrales se fijaron por los autores empíricamente y debían ser $T_1 \in \{4, 8, 12\}, T_2 \in \{15, 25\}, T_3 = 40$ y $T_4 = 50$.

Este es el esquema general para el filtrado basado en $ROM$, sin embargo, en [Abreu96] se mejora dicho esquema porque el valor de reconstrucción ya no es directamente la $ROM$ sino un valor ponderado entre ésta y el píxel analizado, según el grado de degradación sufrido. De esa manera el valor de reconstrucción se obtiene:

$$y(\mathbf{n}) = \alpha_i x(\mathbf{n}) + (1-\alpha_i) m(\mathbf{n}).$$

Dónde los $\alpha_i$ son valores asociados a cada grado de ruido de un determinado pixel. Ese grado de ruido depende de cómo las diferencias $d_i(\mathbf{n})$ superan los diferentes umbrales. Tomando $\mathbf{d}(\mathbf{n})$ como un vector de 4 componentes: $\mathbf{d}(\mathbf{n}) \in \Re^4$; el valor $\alpha_i$ se asocia a la región $i$ de $\Re^4$ en la que se encuentra el vector $\mathbf{d}(\mathbf{n})$. Para ello el espacio $\Re^4$ se particiona



**(a)** **(b)** **(c)**

Figura 5.5: Ejemplo de filtrado ROM con 20 % de ruido impulsivo

en $M$ regiones y a cada una de ellas se le asocia un valor de reconstrucción. En [Abreu96] el valor utilizado para la partición en las simulaciones es $M = 1296$ lo que indica que cada una de las diferencias ($d_i$) puede estar en una de 6 posiciones posibles, de ahí se obtiene $M = 6^4 = 1296$.

Los autores proponen 3 formas de calcular los valores $\alpha_i$ óptimos: Por mínimos cuadrados (Least-squares), algoritmo LMS (Least-Mean Squares) usado en filtrado adaptativo y por último, el que parece dar mejores resultados, que es el diseño recursivo (Recursive Design). Este último utiliza los mínimos cuadrados pero teniendo en cuenta que el filtro se va a aplicar de manera recursiva, es decir, que los valores previamente reconstruidos se van a utilizar para futuras reconstrucciones. Este método necesita de una imagen de entrenamiento de la cual se van extrayendo píxeles que se sabe si son ruidosos o no, se calculan unos valores de $\alpha_i$ y se reconstruye el conjunto de entrenamiento. Después se toma otro conjunto y se actualizan los valores de los $\alpha_i$ con estos nuevos datos. Y este proceso se repite hasta que se terminan todos los conjunto de entrenamiento. Los últimos valores actualizados de los $\alpha_i$ son los que se utilizarán en el proceso de reconstrucción.

En la figura 5.5 se puede ver un ejemplo de eliminación de ruido con este tipo de filtro. Si comparamos la imagen reconstruida con la original veremos que no se produce el efecto de emborronamiento y que el ruido está muy reducido. Esto supone una mejora sobre el filtro de mediana. Sin embargo, si nos fijamos más atentamente en la imagen veremos que el ruido formado por píxeles negros no es eliminado del todo y por tanto la reconstrucción no es total.

La ventaja principal de este método es que se puede aplicar a varios tipos de ruido, incluyendo el gaussiano y el impulsivo con valores aleatorios. Sin embargo, en el ruido "Salt&Pepper" los resultados no son buenos con tasas de ruido medias y altas ([Han97]). Tiene la desventaja, además, de que necesita un entrenamiento que condiciona la reconstrucción posterior, ya que, dependiendo del tipo de imagen usado para el entrenamiento la reconstrucción será mejor o peor.

**Filtro "Minimum-Maximum exclusive mean" (MMEM).** El filtro presentado en [Han97] está especialmente diseñado para su aplicación en imágenes con tasas de ruido altas y sólo para ruido "Salt&pepper". Sin embargo, su sencillez y los buenos resultados que obtiene, merecen su inclusión para comparar con otros métodos propuestos.

La idea básica es utilizar la media en un entorno del píxel en cuestión para obtener los valores de reconstrucción, pero evitando para ello el uso de los píxeles que puedan considerarse ruidosos. En este método se consideran ruidosos aquellos que están cerca del máximo o del mínimo.

El filtro propuesto se basa en el uso de una ventana $W_n(i,j)$ de tamaño $n \times n$ centrada en el pixel $(i,j)$. El filtro se aplica sobre la imagen ruidosa ($g(i,j)$) píxel a píxel y los reemplaza por una estimación sin ruido ($\tilde{g}(i,j)$). Para cada píxel $(i,j)$ se realizan las siguientes operaciones:

1.- Se fija el tamaño de la ventana n=3;

2.- Se buscan el máximo y el mínimo ($g_{MAX}$ y $g_{MIN}$) de los valores de gris dentro de la ventana $W_n(i,j)$.

3.- Se descartan los píxeles $(l,m) \in W_n(i,j)$ que cumplen $\lfloor g(l,m)/4 \rfloor = \lfloor g_{MIN}/4 \rfloor$ o $\lfloor g(l,m)/4 \rfloor = \lfloor g_{MAX}/4 \rfloor$.

4.- Si todos los píxeles dentro de la ventana se descartan entonces, si $n = 3$ se incrementa el tamaño de la ventana a $n = 5$ y se vuelve al punto 2. Si $n = 5$ no se vuelve a incrementar sino que se sigue al siguiente punto.

5.- Se calcula el valor medio ($AVG$) de los píxeles no descartados. Sin embargo si $n = 5$ y todos los píxeles han sido también descartados (tasas de ruido muy altas) el valor de $AVG$ se calcula como la media de los 4 píxeles reconstruidos anteriormente $\tilde{g}(i-1, j \pm 1), \tilde{g}(i-1,j)$ y $\tilde{g}(i, j-1)$.

6.- Si $|AVG - g(i,j)| > 30$ el pixel se considera ruidoso y es reemplazado con $\tilde{g}(i,j) = AVG$, en caso contrario el píxel permanece sin cambios $\tilde{g}(i,j) = g(i,j)$.

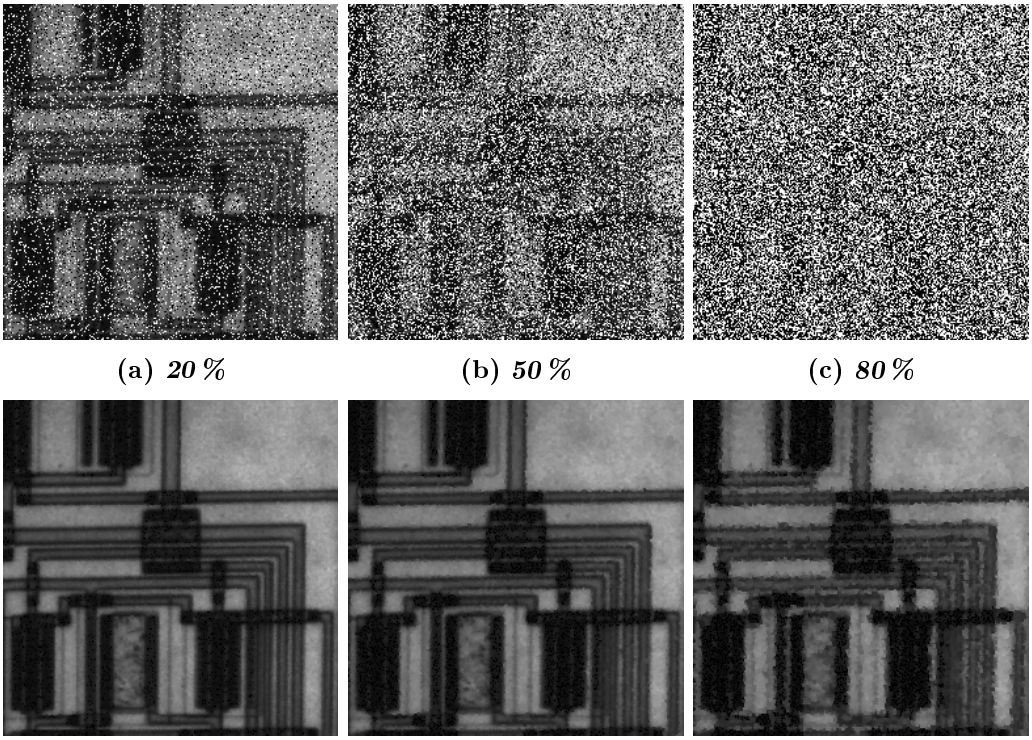
Se puede comprobar que este filtro utiliza el esquema de filtrado condicionado y que, a diferencia de la mayoría de métodos, utiliza la media como valor de reconstrucción. Se puede observar también que es extremadamente simple y, por tanto, muy rápido en la ejecución.

En la figura 5.6 se puede ver un ejemplo de aplicación de este filtro. Las imágenes han sido corrompidas con ruido "Salt&pepper" en tasas del 20, 50 y 80 %. Se puede comprobar visualmente como incluso para tasas altas, como el 50 %, la reconstrucción es buena y para tasas muy altas, como el 80 %, la información de la imagen se recupera de manera importante.

**Filtros de lógica difusa.** Como se ha comentado anteriormente, la lógica difusa [Zadeh65] se ha utilizado para la eliminación del ruido impulsivo con buenos resultados.

**a) Lógica difusa. Introducción.** Aunque la explicación a fondo de dicha lógica escapa al propósito de este documento, a continuación se explican de manera resumida sus características principales. (Una descripción más detallada puede encontrarse en [Jantzen98]).

A diferencia de la lógica clásica, en la que las afirmaciones o son ciertas o son falsas, la lógica difusa amplia las posibilidades permitiendo que haya grados de verdad entre



(a) *20 %* (b) *50 %* (c) *80 %*

(d) ***Recuperada** 20 %* (e) ***Recuperada** 50 %* (f) ***Recuperada** 80 %*

Figura 5.6: Ejemplo de filtrado MMEM con diferentes tasas de ruido.

el falso y el verdadero, o si hablamos de ordenadores, entre el 0 y el 1. Se puede decir, por lo tanto, que en la lógica difusa se permite decir que una afirmación es "más o menos" cierta o falsa. Esta característica hace posible que se pueda hacer procesado con palabras ("Computing with words" según el creador de la lógica difusa Lofti A. Zadeh).

El primer pilar de la lógica difusa es la definición de conjunto ("Sets" en la definición original). La definición clásica de conjunto especifica que los elementos dentro de un determinado conjunto cumplen una o varias características, por ejemplo, "El conjunto de los números enteros positivos entre 1 y 10" está claramente formado por unos elementos que cumplen esa característica de manera inequívoca. Sin embargo, "El conjunto de los números reales mucho más grandes que 1" o "El conjunto de hombres altos" no son conjuntos especificados de manera clara y cerrada sino que su frontera es difusa. Por tanto, en la definición de conjunto difuso no se incluyen los elementos del conjunto sino una función de pertenencia a dicho conjunto, de forma que la transición desde pertenencia a no pertenencia es gradual y no abrupta.

En la figura 5.7 se puede ver una comparación entre la pertenencia a un conjunto vista desde el punto de vista clásico (Salto abrupto en línea continua) o desde el punto de vista de la lógica difusa (Línea discontinua). El grado de pertenencia al conjunto "Hombre alto" crece progresivamente desde 160 cm hasta 180 y se considera que por debajo de 160 la pertenencia es 0 y por encima de 180 es 1.

El siguiente pilar de la lógica difusa son las operaciones sobre los conjuntos, las operaciones difusas y las reglas difusas. Su estudio es complejo pero son las que dan potencia

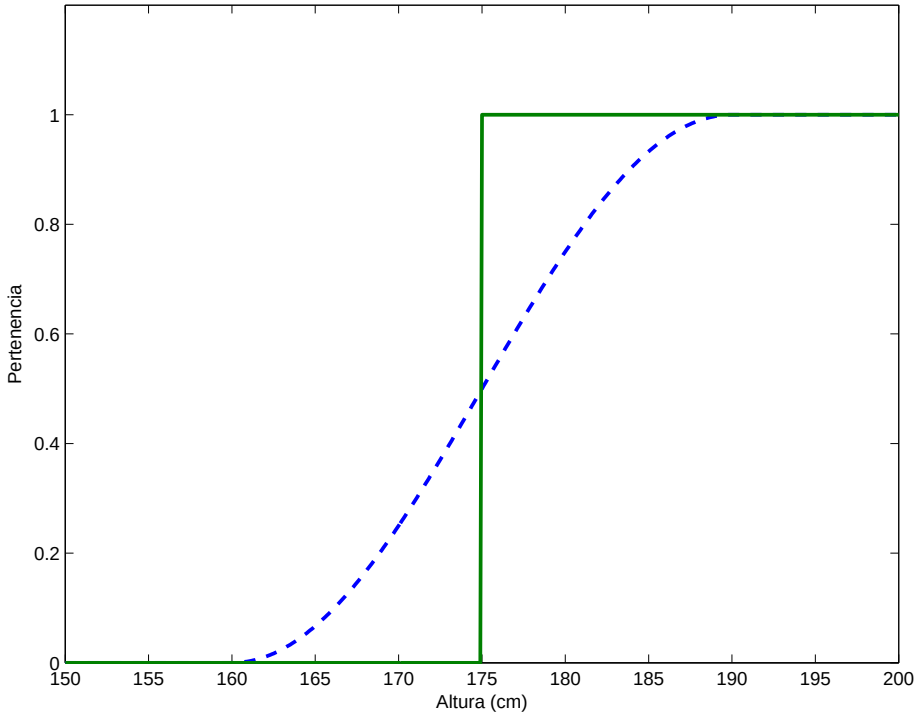



Figura 5.7: Ejemplo de función de pertenencia para el conjunto "Hombre alto".

a la lógica difusa en sí, ya que permiten el procesado lingüístico. Una de las reglas que más se utilizan en el procesado de las imágenes para la eliminación del ruido es la "regla de inferencia" en la que se llega a una conclusión difusa a partir del más o menos cumplimiento de una condición. Este tipo de reglas se enuncian:

$$\text{Sí } x \text{ es } \mathfrak{A}, \text{ entonces } y \text{ es } \mathfrak{B},$$

dónde $\mathfrak{A}$ y $\mathfrak{B}$ son conjuntos difusos que pueden estar definidos en universos distintos. Ejemplos de estas reglas pueden ser:

1.- Si está oscuro, entonces conduce lentamente.

2.- Si el tomate está rojo, entonces está maduro.

3.- Si es temprano, entonces puedo estudiar.

En todos estos casos el resultado final no será una acción "todo o nada" sino un resultado que nos dirá el grado de cumplimiento de la regla. Por ejemplo, en la regla número 3, "temprano" es un conjunto de horas a lo largo del día con una función de pertenencia que será máxima a las 8 de la mañana y que irá decreciendo hasta 0 de madrugada. Si son las 8 horas, el grado de pertenencia será 1 y por tanto podré estudiar con grado 1. Sin embargo, a las 20 horas el grado de pertenencia será 0.5 y sólo podré estudiar con grado 0.5.

b) **Filtro DS-FIRE (Dual Step-Fuzzy Inference Ruled by Else-action).** Dentro de los métodos de eliminación de ruido impulsivo basados en lógica difusa se suele referenciar en la literatura el conocido como filtro "DS-FIRE" [Russo96a, Russo96b].

| | | 1 | 2 | 3 | | |
|---|---|---|---|---|---|---|
| | | 4 | 5 | 6 | | |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | **0** | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| | | 27 | 28 | 29 | | |
| | | 30 | 31 | 32 | | |

Figura 5.8: Ventana en cruz $7 \times 7$.

Este filtro realiza el filtrado en dos pasos, estos dos pasos no son de detección y sustitución sino de sustitución gruesa y fina. Por tanto, este método no se puede englobar en el tipo de los clasificadores. El objetivo final de este método es la obtención de un término de corrección, calculado para cada píxel, de forma que se reduzca el nivel de ruido de la imagen. Si $x(\mathbf{n}) = x_0$ es el valor de un pixel de una imagen ruidosa en la posición $\mathbf{n} = [n_1, n_2]$, el método dará un valor de salida $y(\mathbf{n}) = x(\mathbf{n}) + \Delta y(\mathbf{n})$ donde $\Delta y(\mathbf{n})$ es el término de corrección obtenido por el procesado difuso. El filtro opera en una ventana alrededor del píxel en cuestión, en [Russo96a] la ventana usada es de $3 \times 3$ mientras que en [Russo96b] la ventana es más complicada y se define como una cruz de $7 \times 7$ como se puede ver en la figura 5.8. El método es básicamente el mismo en los dos casos, aunque al usar una ventana de $7 \times 7$ aparecen un número mayor de reglas difusas que evaluar, 50 frente a 26. Los píxeles de entrada se llaman $x_j$ y los de salida $y_j$, donde los subíndices se corresponden a los índices que se muestran en la figura 5.8. Una característica de este filtro es que es recursivo, es decir, los píxeles dentro de la ventana que ocupan posiciones anteriores al píxel central han sido ya procesados.

Las variables de entrada al procesado difuso no son directamente los píxeles de la ventana sino las diferencias del píxel central con el resto de los de la ventana:

$$\Delta x_j = \begin{cases} y_j - x_0 & j = 1, \ldots, 16 \\ x_j - x_0 & j = 17, \ldots, 32 \end{cases} . \tag{5.2}$$

Con todos esos elementos se puede proceder a la eliminación del ruido realizando los siguientes pasos:

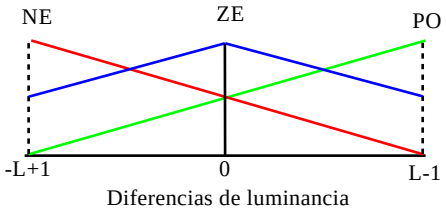



Figura 5.9: Funciones de pertenencia triangulares para los conjuntos difusos PO (Positivo), ZE (Cero), NE (Negativo).

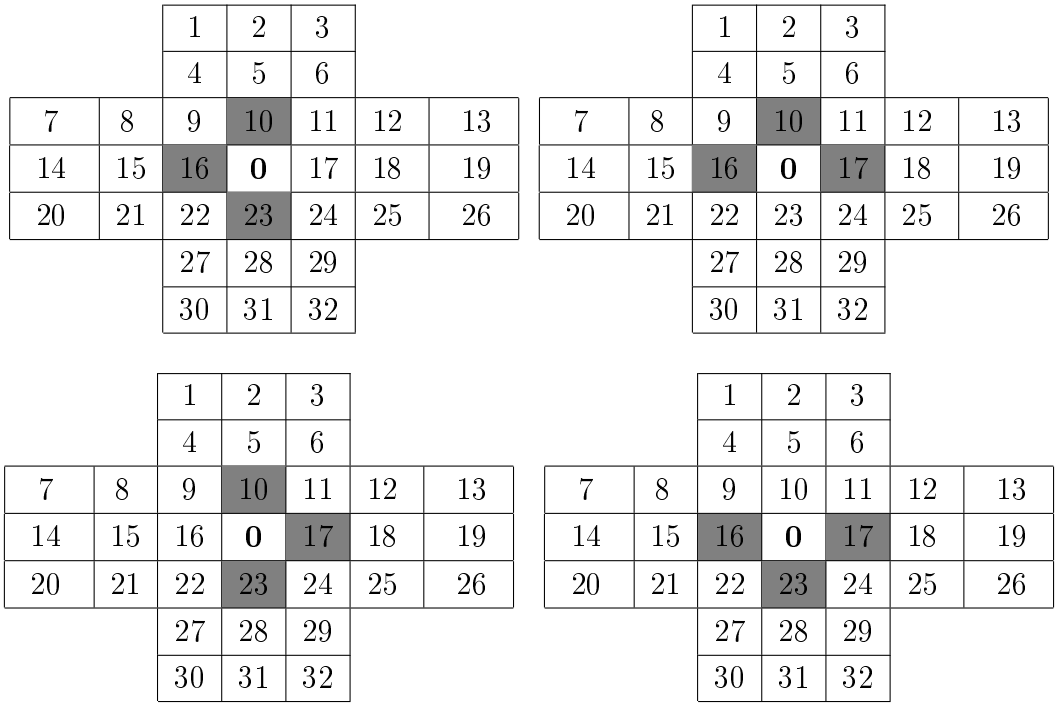



Figura 5.10: Subconjuntos definidos para los subíndices $I_1$, $I_2$, $I_3$ e $I_4$.

1.- Se eligen unos subconjuntos de las diferencias de luminancia calculadas. Los subconjuntos se definen por los subíndices de las diferencias (5.2). Por ejemplo, el subconjunto $I_1 = \{10, 16, 23\}$ se refiere a las diferencias del píxel central con los que ocupan esas posiciones en la ventana en cruz de $7 \times 7$. En la figura 5.10 se representan gráficamente los cuatro subconjuntos más simples. En [Russo96b] a esos cuatro más simples se le añaden 21 más complejos para mejorar los resultados, con lo cual el número total de subconjuntos es de 25.

2.- Se calculan los grados de pertenencia de dichos subconjuntos tanto al conjunto PO como al NE obteniendo los siguientes valores:

$$\begin{aligned} \lambda_1 &= MAX\{MIN\{PO(\Delta x_j); j \in I_i\}; i = 1, \ldots, 25\}, \\ \lambda_2 &= MAX\{MIN\{NE(\Delta x_j); j \in I_i\}; i = 1, \ldots, 25\}. \end{aligned} \tag{5.3}$$

3.- Con esos valores calculados se obtiene una primera aproximación al valor de reconstrucción según la ecuación:

$$\Delta y' = (L-1)\frac{(\lambda_1 - \lambda_2)}{(\lambda_1 + \lambda_2 + \lambda_0)}, \tag{5.4}$$

donde se elige el valor $\lambda_0 = 1 - (\lambda_1 + \lambda_2)$ y, por tanto:

$$\Delta y' = (L-1)(\lambda_1 - \lambda_2). \tag{5.5}$$

4.- Los puntos anteriores se corresponden con el primer paso del algoritmo. El siguiente paso es de refinamiento del valor obtenido con el objetivo de preservar

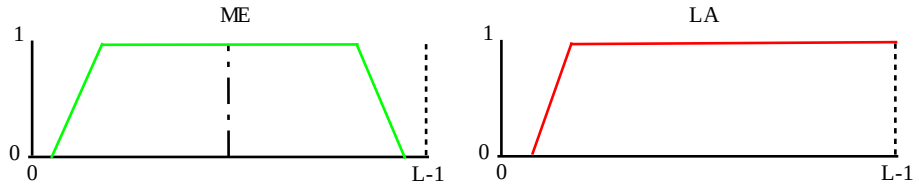



Figura 5.11: Funciones de pertenencia para los conjuntos difusos ME (Medio) y LA (Grande).

los detalles finos y las texturas. Para ello se reducen los valores de reconstrucción obtenidos que sean pequeños o correspondan a píxeles en los valores medianos de la escala de grises. De forma que el valor final de reconstrucción es:

$$\Delta y = \Delta y' MAX\{LA(|\Delta y'|), 1 - ME(x_0)\}, \tag{5.6}$$

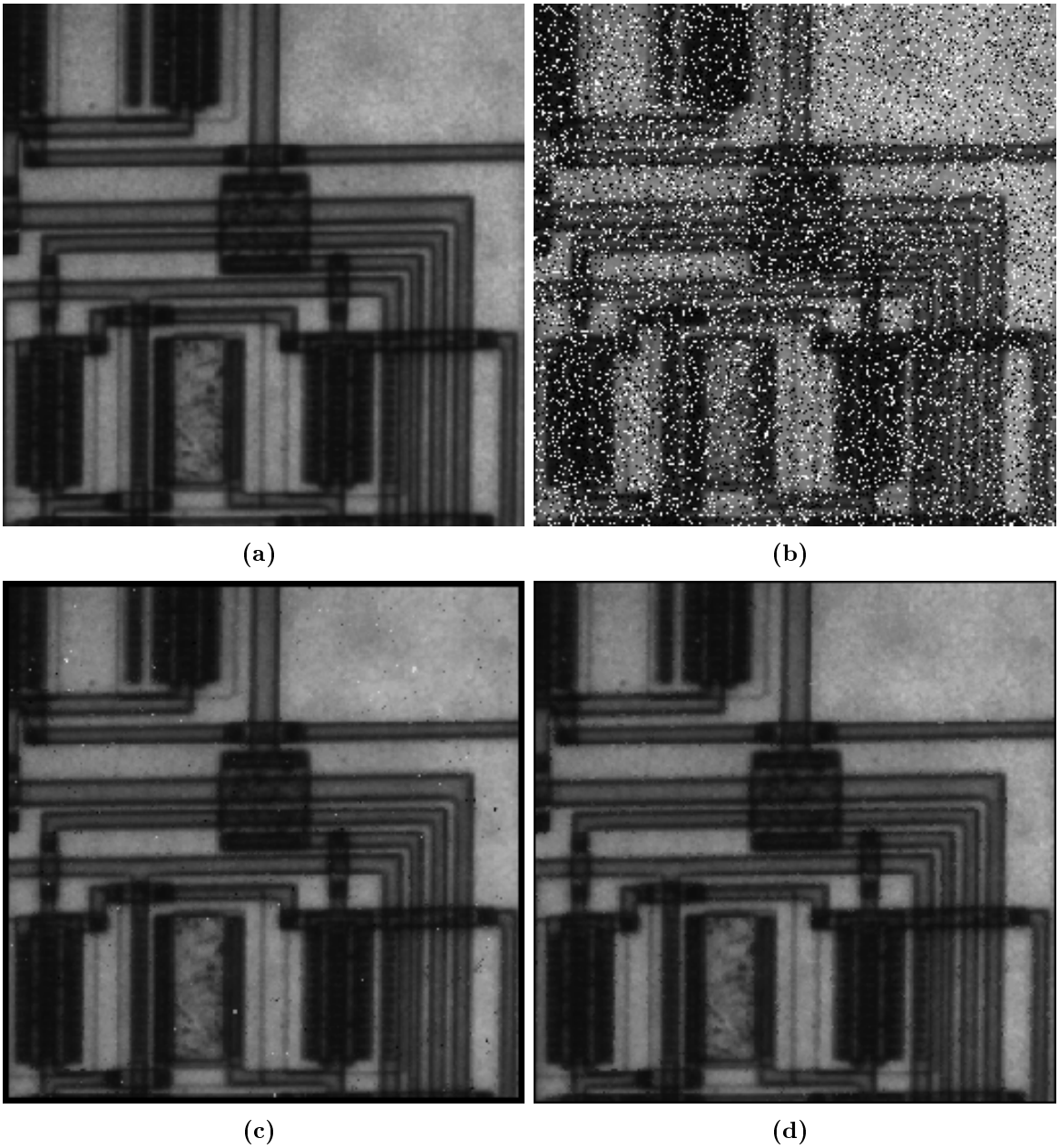
donde LA (Grande) y ME (Medio) son dos conjuntos difusos cuyas funciones de pertenencia se muestran en la figura 5.11. Podemos ver que este segundo paso no afecta a valores de corrección que son *grandes (LA)* o se corresponden con valores de píxeles que *no son medianos (ME)*.

En la figura 5.12 puede verse un ejemplo de reconstrucción utilizando este método en una imagen con el 20 % de ruido impulsivo.

**c) Filtro FIDRM (Fuzzy Impulse noise Detection and Reduction Method).** Debido a los buenos resultados de los filtros de lógica difusa estos han sido estudiados y mejorados añadiéndoles complejidad y nuevas prestaciones. Recientemente en [Schulte06b, Schulte06a] se ha propuesto un filtro basado en la lógica difusa que se aplica en dos fases: una de detección y otra de filtrado. La fase de detección usa reglas difusas para determinar si un píxel está afectado por ruido impulsivo. Estas reglas difusas deciden el grado de afectación por el ruido distinguiendo entre píxeles pertenecientes a los bordes naturales de la imagen y aquellos debidos al ruido añadido, para eso se basan en el concepto de "Valores de gradiente difusos" (Fuzzy Gradient Values). En esta fase no sólo se realiza la detección del ruido sino que se calculan varios parámetros referidos a este y que después serán usados en la fase de filtrado.

La fase de filtrado se centra, por tanto, sólo en los píxeles realmente ruidosos, mejorando la calidad de la reconstrucción. El filtrado se basa en el cálculo de una función de pertenencia difusa llamada "Más o menos ruido impulsivo", a partir de la cual se obtiene el valor de reconstrucción. Esta función difusa se va modificando de manera recursiva cuando es necesario, si la cantidad de ruido detectada es muy grande. Para esta aplicación recursiva se parte de la imagen reconstruida en la que ya se han sustituido algunos píxeles y se va ampliando la ventana alrededor del píxel en cuestión desde la inicial de $3 \times 3$. Este proceso recursivo finaliza cuando el número de píxeles ruidosos detectados es nulo, permanece fijo de una iteración a otra o es suficientemente pequeño.

Los resultados de este método son mejores que los del filtro DS-FIRE tanto en calidad de reconstrucción como en tiempo de ejecución. En la figura 5.12 se puede ver un ejemplo de reconstrucción con este filtro comparado con el DS-FIRE.



(a) (b)

(c) (d)

Figura 5.12: Ejemplo de filtrado de lógica difusa. (a) Imagen Original. (b) Imagen con 20 % de ruido. (c) Imagen recuperada con el método DS-FIRE. (d) Imagen recuperada con el método FIDRM

## 5.2. Métodos propuestos

Una vez analizado el estado del arte en cuanto a métodos de eliminación de ruido impulsivo, queda claro que un esquema en el que haya una detección previa de los píxeles de la imagen ruidosos seguido de una reconstrucción es el método más eficiente. Es eficiente en tiempo y en resultados obtenidos.

Las propuestas realizadas siguen el esquema de barrido de la imagen usando una ventana de un tamaño determinado (generalmente de $3\times3$) alrededor del píxel en cuestión y de la que se extraerá un vector que habrá de ser etiquetado.

En los siguientes puntos se proponen una serie de técnicas que usan clasificadores ([Gómez-Moreno03]) y métodos de regresión basados en SVM ([Gómez-Moreno01a]), o en RBF como alternativa más rápida. Los resultados obtenidos comparados con algunos de los métodos más representativos del estado del arte mejoran en algunos aspectos y muestran una nueva vía de investigación en este tema.

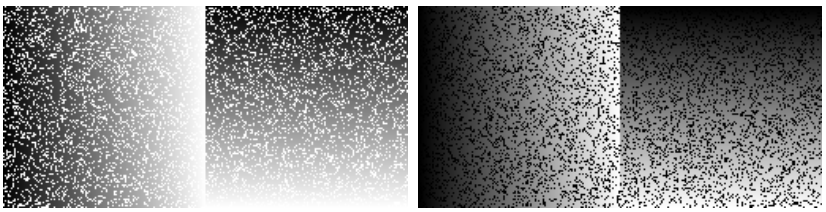
### 5.2.1. Detección de ruido impulsivo

La detección del ruido impulsivo será, por tanto, la clave para el posterior reemplazo de los píxeles clasificados como ruidosos. Para dicha detección hay que entrenar el clasificador con imágenes en las que el ruido esté previamente localizado, de forma que se pueda discernir entre píxeles ruidosos y no ruidosos. En este punto se propone el uso de imágenes de entrenamiento sintéticas, en las que tanto el tamaño de las mismas como el porcentaje de ruido sea definido previamente. Ya en el capítulo de detección de bordes se usaron este tipo de imágenes con buenos resultados.

**Imágenes de entrenamiento.** En la figura 5.13 se muestra el tipo de imagen elegido para el entrenamiento de clasificación. Está dividida en dos sub-imágenes, la izquierda con una gradación de grises vertical y la derecha con una gradación horizontal. Esa gradación va del negro al blanco pasando por los valores intermedios. Esos valores intermedios dependerán del tamaño de sub-imagen elegido, el salto entre valores de gris es de $\frac{256}{\text{Tamaño}}$ y el tamaño debería ser un múltiplo de 2. Imágenes más grandes tendrán una mayor gama de valores de gris y el entrenamiento será más cercano a la realidad. Las dos sub-imágenes se colocan de esa manera para que haya al menos un borde en la imagen y que éste tenga distintos niveles de gris. Las pruebas realizadas inicialmente daban buenos resultados con esa configuración aunque otras más complejas puedan mejorarlos a costa de complicar el entrenamiento.

A las imágenes de entrenamiento así formadas se les añade ruido blanco o negro en el porcentaje elegido (Figuras 5.13(a) y 5.13(b)) y se leen siguiendo el esquema de enventanado elegido. Obviamente los píxeles del ruido son conocidos pues en la generación del mismo se van guardando las posiciones donde se encuentran.

Como se puede observar, el entrenamiento propuesto separa los dos tipos de ruido (píxeles blancos y píxeles negros) porque los resultados obtenidos usando un modelo común producían una mayor complejidad de los clasificadores pero sin mejora en la clasificación. Por tanto, se obtendrán dos modelos de entrenamiento, uno para la detección del ruido blanco y otro para la detección del ruido negro, pues de esta manera se optimiza el proceso.



(a) (b)

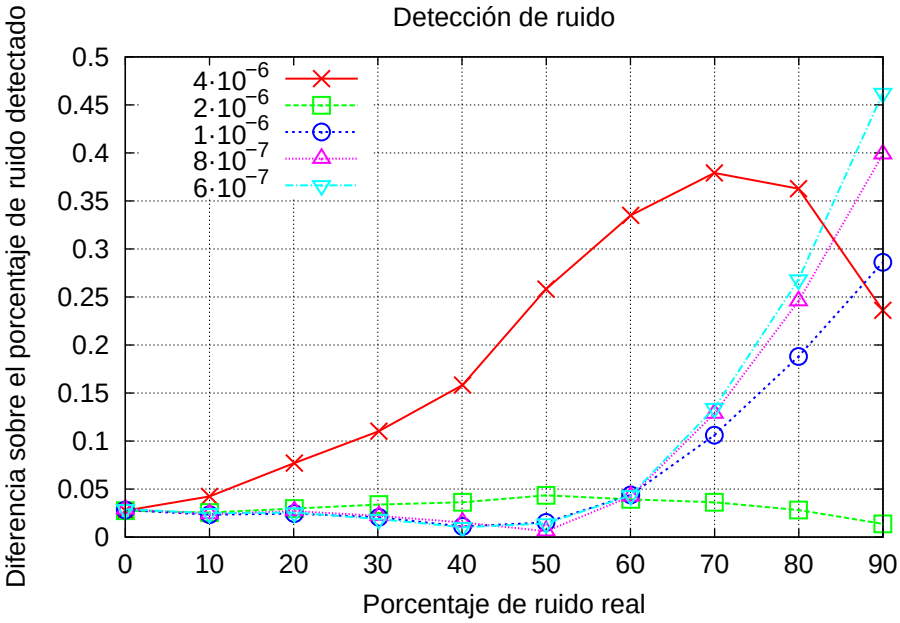
Figura 5.13: Ejemplo de imágenes $128 \times 128$ de entrenamiento para la detección de ruido. Ambas imágenes tienen un 20 % de ruido añadido. (a) Imagen para detección de ruido blanco; (b) Imagen para detección de ruido negro.

**Parámetros del entrenamiento.** En el entrenamiento con los datos obtenidos de estas imágenes hay que fijar los parámetros del kernel usado y la constante de regularización. Tras unas pruebas iniciales se decidió el uso del kernel gaussiano por obtener los mejores resultados con un número reducido de vectores soporte, por tanto, el parámetro a fijar será la constante $\gamma$ que controla la anchura del kernel. La constante de regularización se fijó de manera experimental, comprobando que su efecto en la clasificación es reducido, pero sí es importante en el número de vectores soporte obtenido, con esos datos se fijó un valor de $C = 1000$, ya que valores inferiores aumentaban el número de vectores soporte y valores superiores no suponían mejora alguna.

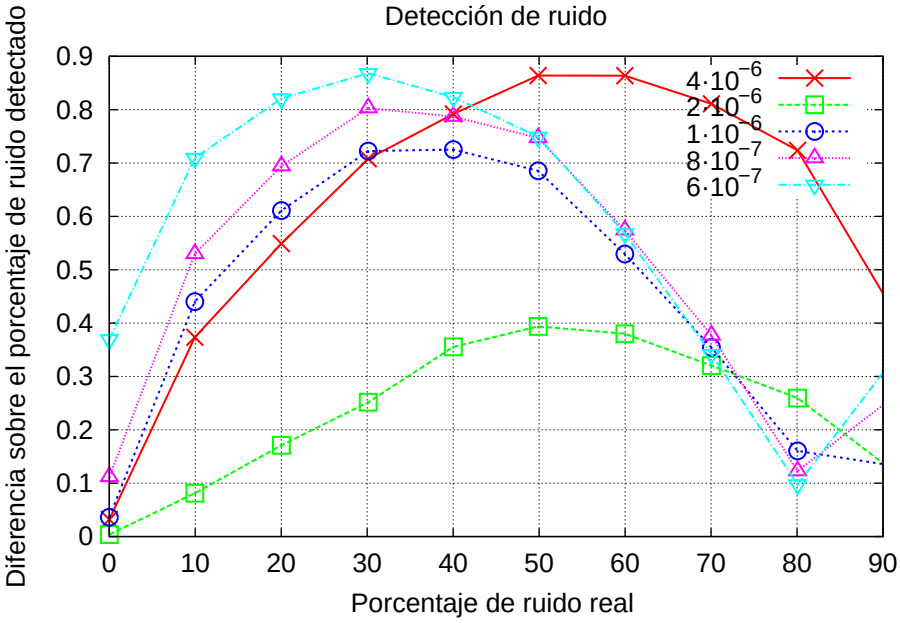
Para fijar el valor de $\gamma$ se realizaron distintas pruebas de clasificación en distintas imágenes y se observó que el valor óptimo era $\gamma = 2 \cdot 10^{-6}$. En estas pruebas se utilizaron imágenes de entrenamiento de tamaño 32 con un porcentaje de ruido de entrenamiento del 40 %. En la figura 5.14 se muestran resultados para las imágenes Lenna y Albert. Estas gráficas muestran la diferencia entre el porcentaje de ruido detectado y el realmente presente en función de este último. Se puede apreciar cómo el valor de $\gamma$ en el que la diferencia es más estable es el comentado anteriormente. En la gráfica 5.14(a) se observa que para determinados valores de ruido hay valores de $\gamma$ ligeramente mejores pero no tan estables como el elegido, es decir, crecen a partir de un valor de ruido. Algo similar ocurre con la gráfica 5.14(b) aunque en este caso la elección se produce no ya por la estabilidad sino por que los valores de clasificación son mejores.

En todo caso los parámetros de entrenamiento se pueden adecuar a la imagen o tipo de imágenes que se van a utilizar y a las cantidades de ruido que se esperan encontrar. Como este trabajo pretende ser aplicable a cualquier imagen y para cualquier cantidad de ruido se ofrecen los valores de entrenamiento mejores para la mayoría de los casos, pero el método y las gráficas utilizadas también podrían usarse para casos particulares.

**Comparación en diferentes imágenes.** Una vez fijado el valor para los parámetros de entrenamiento se debe comprobar el efecto de la variación de los distintos factores asociados a la detección. El primero de ellos es la imagen sobre la que se efectúa la detección. Las figuras 5.15 y 5.16 muestran algunos ejemplos de los resultados que se pueden obtener dependiendo del tipo de imagen sobre el que estemos trabajando. En la figura 5.15 aparecen gráficas de diferencia de ruido detectado y real sobre 4 imágenes distintas y con distintos entrenamientos. Aunque los resultados cambian con el entrenamiento, se comprueba que los resultados correspondientes a una misma imagen son similares y que




**(a) *Resultados para Lenna***




**(b) *Resultados para Albert***

Figura 5.14: Resultados de clasificación para varios valores de $\gamma$. Se representa la diferencia entre el ruido detectado y el realmente presente en la imagen respecto al porcentaje de ruido real.

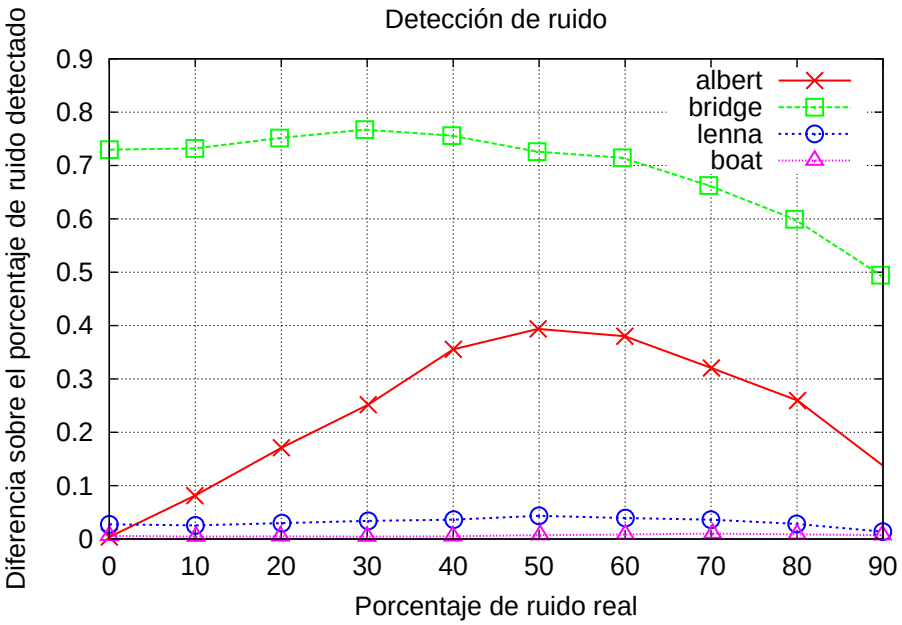
la imagen sobre la que se trabaja es importante a la hora de obtener resultados. Imágenes como Boat o Lenna obtienen mejores resultados que Albert o Bridge aunque hay que resaltar que en todo caso la diferencia entre el ruido real y el realmente detectado no es mayor del 0.8 %.

La figura 5.16, en la que se muestran resultados gráficos de detección, puede dar una idea del porqué de las diferencias entre las imágenes. En esta figura, junto con la imagen ruidosa, se muestra el ruido detectado marcado con un punto blanco, de esta manera se puede comprobar donde se producen exactamente los errores. Las imágenes Boat y Lenna ofrecen buenos resultados y así aparece en la figura, lo que caracteriza estas imágenes es que son escenas donde predominan los grises y no hay zonas grandes totalmente blancas o negras. Sin embargo, en Albert o Bridge sí aparecen estas zonas. En Albert la parte de la camisa produce falsas detecciones porque es uniformemente blanca, de forma que cuando se añade ruido negro hay píxeles de esa zona que, a su vez, se detectan como ruido blanco. Sin embargo, si tomamos la imagen sin ruido no se producen ese tipo de falsas clasificaciones, ya que no se detecta ruido alguno. En la imagen Bridge, aparece una zona uniformemente negra, debida a un error de escaneo, en la parte inferior derecha. Esta zona produce un efecto similar al comentado anteriormente para Albert. Por tanto, la clasificación de píxeles ruidosos funcionará mejor en imágenes con una gama amplia de grises pero en las que no aparezcan zonas uniformemente blancas o negras.

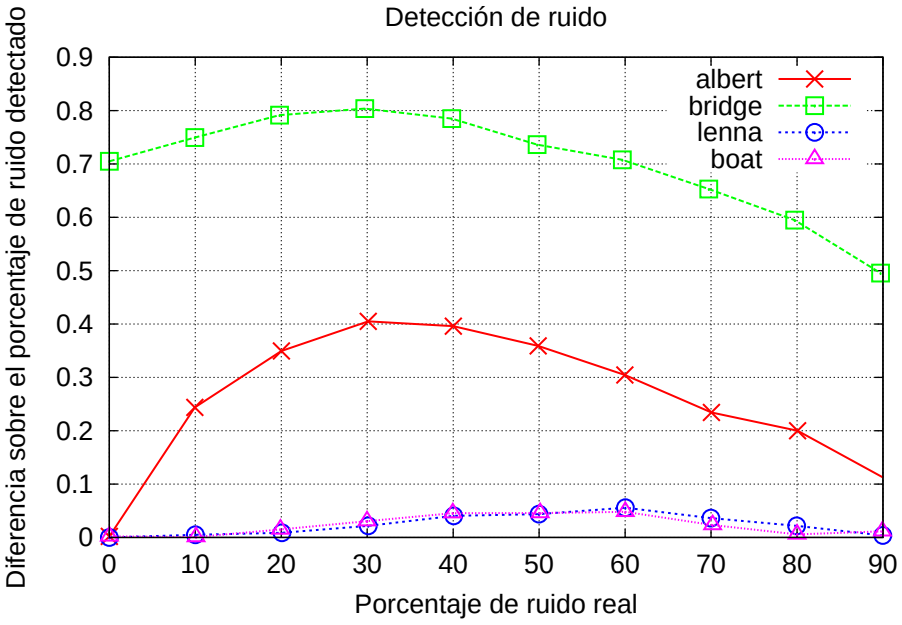
**Efecto del ruido y el tamaño de las imágenes de entrenamiento.** Al diseñar las imágenes de entrenamiento se comentó que éstas pueden variar en tamaño y en el porcentaje de ruido añadido. Estos dos parámetros son cruciales a la hora de definir la calidad del entrenamiento así como la posterior rapidez o no en la ejecución de la clasificación. Un mayor tamaño implica un mayor número de valores de gris, un número mayor de píxeles ruidosos (aunque el porcentaje sea el mismo) y un número mayor de vectores para usar en el entrenamiento. Todo ello hace que a mayor tamaño, mejor y más cercano a la realidad sea el entrenamiento aunque a costa de crecer en número de vectores soporte usados. La cantidad de ruido añadida al entrenamiento influye en que a mayor ruido más ejemplos de píxeles ruidosos aparecen y, por tanto, el entrenamiento será más completo. Sin embargo, la cantidad de ruido no debe ser muy grande porque si no aparecerán grupos de píxeles ruidosos sin ningún píxel gris alrededor que, en lugar de mejorar, empeorarán el entrenamiento al generar zonas uniformes blancas o negras y no píxeles aislados que es lo que se busca.

En las figuras 5.17 y 5.18 se muestran distintas gráficas en las que se puede apreciar la variación del error cometido en la clasificación para distintos niveles de ruido y tamaño de la imagen de entrenamiento. En la figura 5.17 aparecen 3 gráficas con resultados en función del ruido añadido (30, 40 ó 50 %). Cada una de esas gráficas utiliza un tamaño distinto de imagen. Los mejores resultados se obtienen con un ruido añadido del 50 % aunque para un nivel del 40 % también se obtienen buenos resultados. Hay que destacar que el número de vectores soporte aumenta al aumentar el ruido entrenamiento y el tamaño de la imagen pues también se utilizan para entrenar un número mayor de vectores distintos. Como. ejemplo, para el caso de de un ruido de entrenamiento del 40 %, el tamaño 32 produce 49 vectores, el 64 92 y el 128 261. Esto supone un incremento de tiempo de ejecución.

Para el caso de la figura 5.18 no hay una conclusión clara pues no hay un tamaño de imagen que produzca buenos resultados en todos los casos y puede darse el caso de la figura




**(a)** ***Entrenamiento de 32×32 y 40 % de ruido***




**(b)** ***Entrenamiento de 64×64 y 50 % de ruido***

Figura 5.15: Resultados comparados de detección de ruido en 4 imágenes distintas. Se representa la diferencia entre el ruido detectado y el realmente presente en la imagen respecto al porcentaje de ruido real.

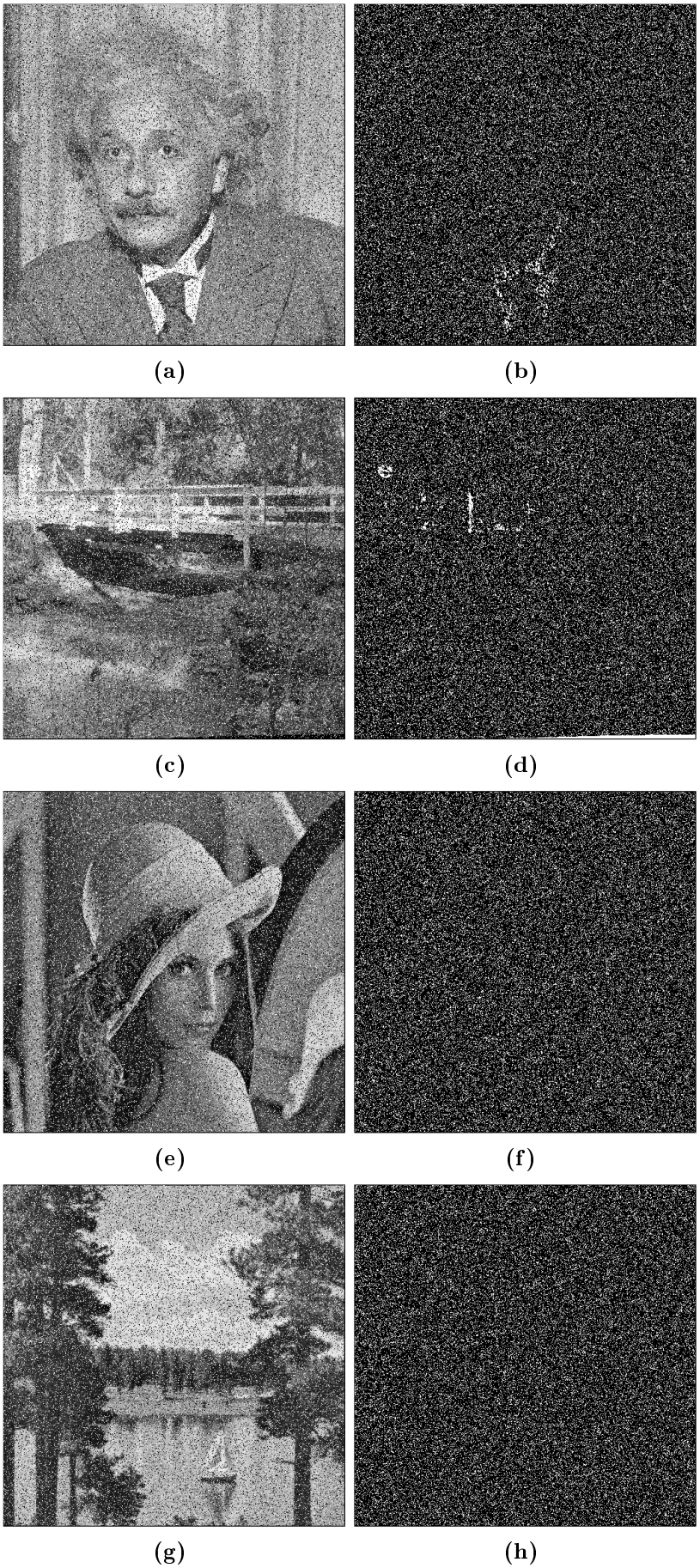
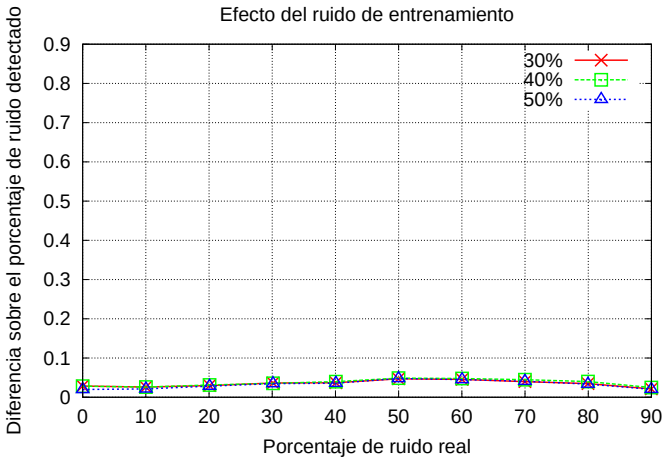
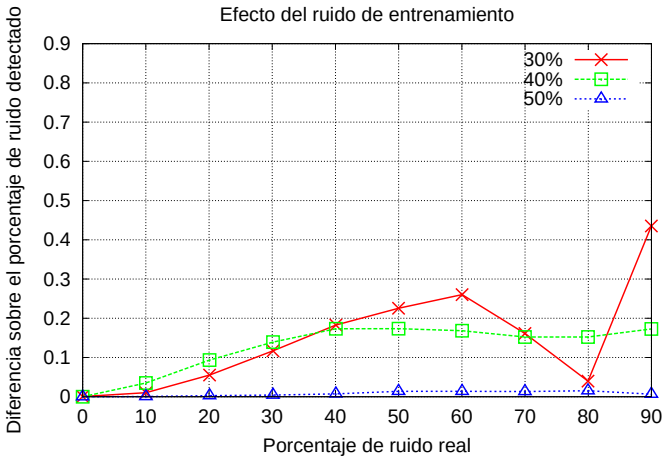


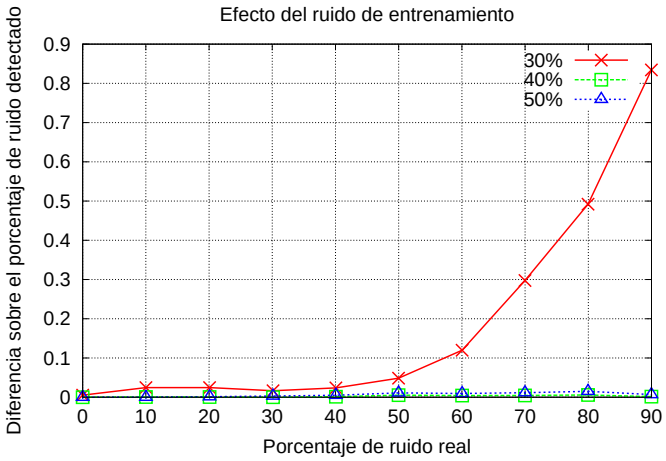

Figura 5.16: Resultados de clasificación de ruido usando un modelo de $32 \times 32$ y 40 % de ruido. (a),(c),(e),(g) Imágenes con 20 % de ruido añadido; (b),(d),(f),(h) Ruido realmente detectado.




**(a)** ***Tamaño 32***

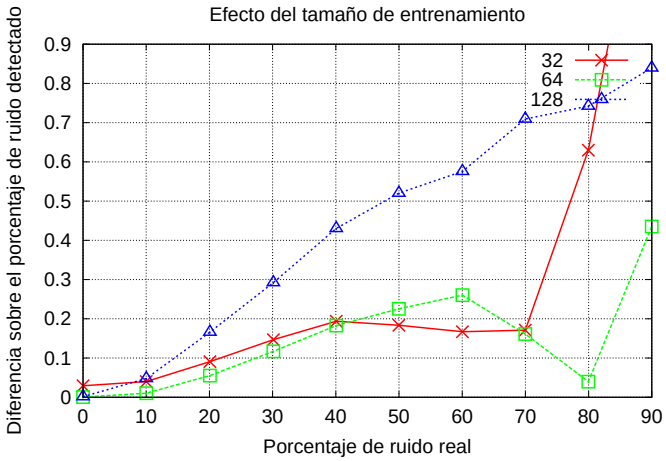



**(b)** ***Tamaño 64***

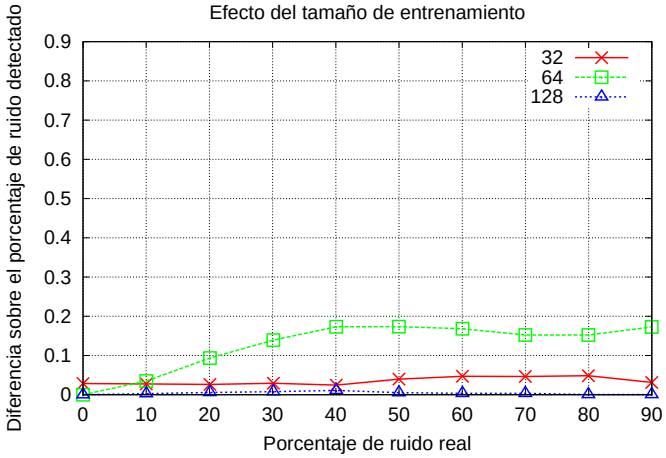



**(c)** ***Tamaño 128***

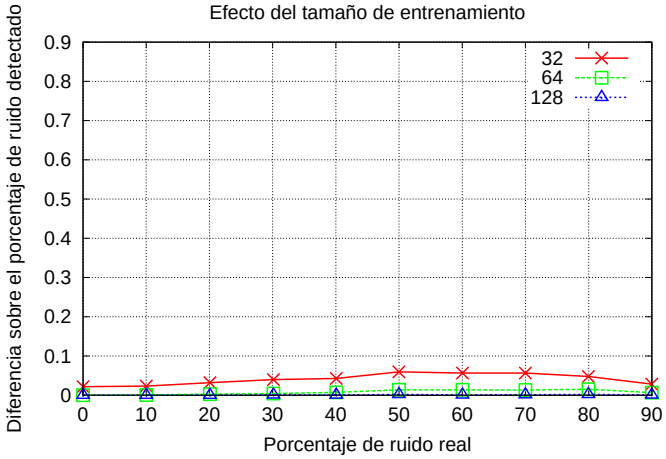
Figura 5.17: Resultados de detección para distintos ruidos de entrenamiento sobre la imagen Lenna. Se presentan distintos tamaños de la imagen de entrenamiento.




**(a) *Ruido 30***




**(b) *Ruido 40***




**(c) *Ruido 50***

Figura 5.18: Resultados de detección para distintos tamaños de entrenamiento sobre la imagen Lenna. Se presentan distintos ruidos de entrenamiento.

5.18(c) con un 50 % de ruido añadido donde los resultados son buenos independientemente del tamaño de imagen elegido.

Obviamente, se comprueba que los dos parámetros son importantes en la clasificación aunque después de examinar los resultados obtenidos parece claro que tiene una influencia más clara el ruido añadido que el tamaño de las imágenes de entrenamiento. Sin embargo, hay que llegar a un compromiso entre calidad de reconstrucción y tiempo de ejecución ya que los modelos más complejos con tamaño de 128 y ruido 50 % implican también un número mayor de vectores.

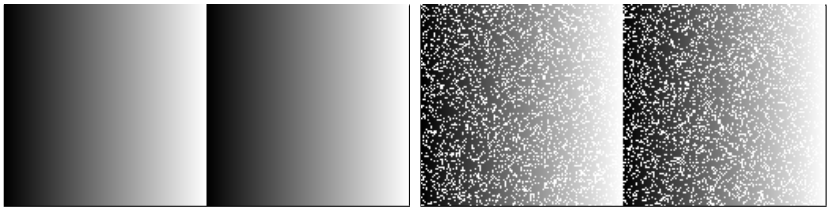
### 5.2.2. Regresión de ruido impulsivo

En este apartado nos centraremos en el problema de la recuperación de los valores ocupados por píxeles ruidosos; para ello, la única información disponible son los píxeles adyacentes. Debemos, por tanto, obtener un valor desconocido a partir de otros que pueden ser ruidosos. Este es el problema tipo que se puede tratar usando herramientas de regresión. En nuestro caso se van a usar SVMs por razones similares a las presentadas en la clasificación, buena generalización y resultados aceptables con una información de entrenamiento reducida.

Partimos, por tanto, de la imagen ruidosa y la información son los valores asociados a los píxeles alrededor del que hay que obtener. Así, el vector de entrada estará formado por los valores de gris en una ventana de $n \times n$ pero quitando el píxel central pues se asume que es ruidoso y no se puede utilizar. Ventanas más grandes usarán más información aunque darán valores más alejados del deseado y ventanas más pequeñas se acercarán más al valor real. Pero si el ruido es grande, todos los píxeles dentro de ella pueden ser ruidosos. Para las pruebas se ha optado por un valor de $3 \times 3$ porque, aún siendo pequeño, da buenos resultados y reduce la complejidad.

**Imágenes de entrenamiento.** El primer paso debe ser el de entrenar la SVM para que puedan obtenerse valores adecuados de sustitución para los píxeles ruidosos. Vistos los buenos resultados obtenidos en puntos y capítulos anteriores con el entrenamiento mediante imágenes sintéticas se decidió su uso para esta aplicación. Para simplificar al máximo el problema de la regresión se planteó que sólo se buscarán los valores de sustitución de los píxeles de ruido blancos pues, en pruebas iniciales, la mezcla de ambos ruidos no da los resultados esperados. El hecho de recuperar valores tan dispares como 0 y 255 producía que los valores de reemplazo no fueran los correctos sino que estaban alejados de los píxeles no ruidosos adyacentes. Por ello, el método de regresión desarrollado se aplicará sólo sobre píxeles de ruido blancos. Los píxeles de ruido negros serán sustituidos por blancos cuando se realice la clasificación previa.

En la figura 5.19 tenemos un ejemplo de las imágenes utilizadas en este entrenamiento. Aparecen valores de gris en una escala de 0 a 255 y un borde central. La variedad de valores de gris dependerá del tamaño de la imagen como en el entrenamiento para la clasificación. En la figura 5.20 pueden verse algunas líneas de un fichero usado para el entrenamiento de regresión. Cada línea representa un vector de entrenamiento y su valor asociado. El primer valor es el del píxel original no ruidoso y, por tanto, el valor deseado. Los otros son las componentes del vector siendo el primero el valor correspondiente al píxel superior izquierdo en la ventana $3 \times 3$ y el último el inferior derecho.



(a) (b)

Figura 5.19: Ejemplo de imagen $128 \times 128$ de entrenamiento con un 20 % de ruido añadido para la regresión de ruido. (a) Imagen no ruidosa tomada como modelo. (b) Imagen ruidosa con un 20 % de impulsos blancos.

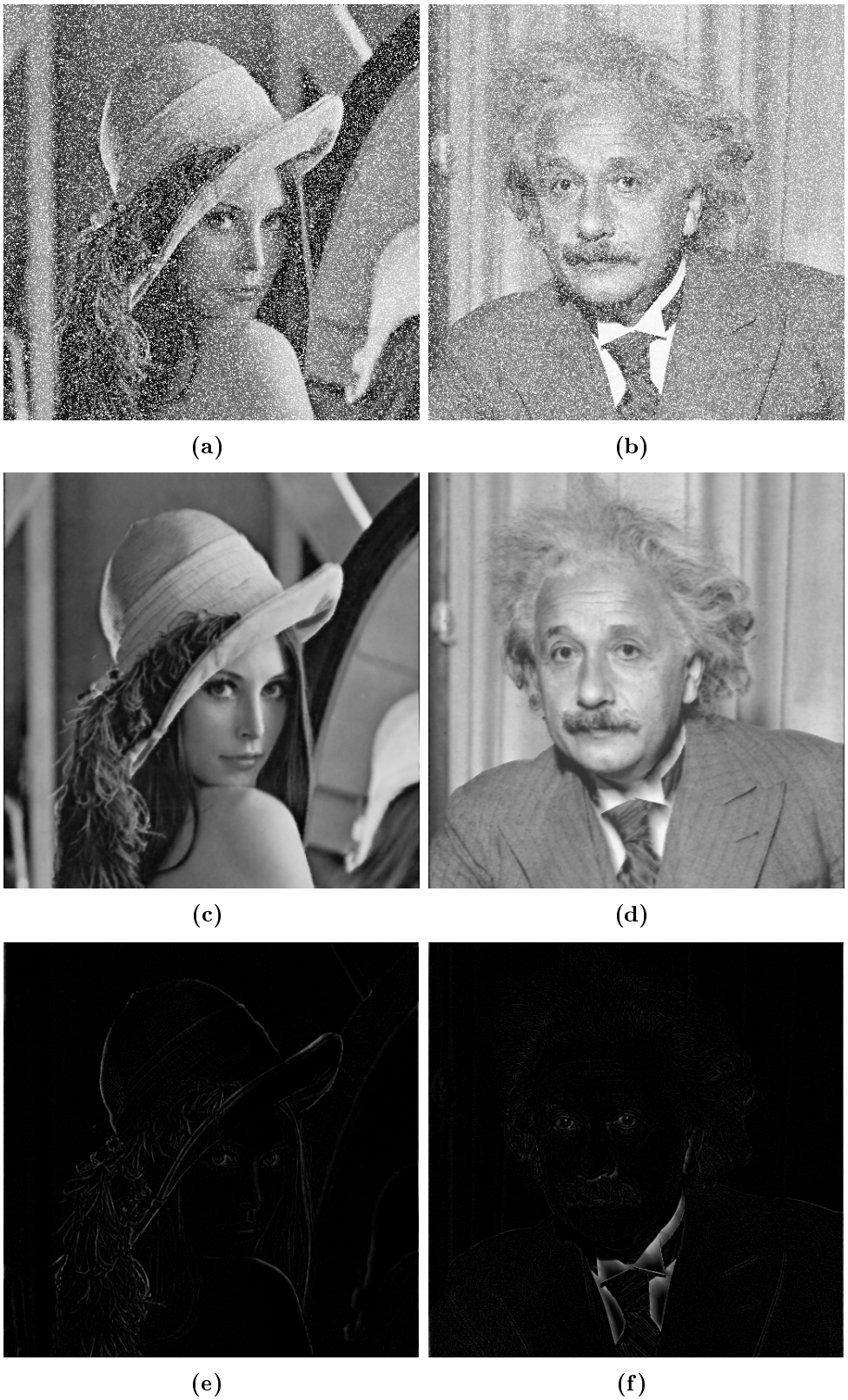
```
 4 1:255 2:255 3:8 4:255 5:255 6:0 7:4 8:8
 8 1:255 2:8 3:12 4:255 5:12 6:4 7:8 8:12
12 1:8 2:12 3:16 4:255 5:16 6:8 7:12 8:16
16 1:12 2:16 3:20 4:12 5:20 6:12 7:16 8:255
20 1:16 2:20 3:255 4:16 5:255 6:16 7:255 8:24
24 1:20 2:255 3:28 4:20 5:28 6:255 7:24 8:255
28 1:255 2:28 3:32 4:255 5:32 6:24 7:255 8:32
32 1:28 2:32 3:36 4:28 5:36 6:255 7:32 8:255
```

Figura 5.20: Ejemplo de vectores de entrenamiento para la regresión.

Es un entrenamiento sencillo y configurable en cuanto al tamaño (y, por tanto, valores de gris) y el ruido añadido. Sin embargo, los resultados obtenidos son buenos con unos parámetros de entrenamiento básicos, por ejemplo, un kernel gaussiano con una $\gamma = 3 \cdot 10^{-5}$, $C = 1000$ y $\varepsilon = 10^{-3}$. En la figura 5.21 se muestran dos ejemplos en los cuales no se ha realizado la clasificación sino la aplicación directa de la regresión. El efecto deseado de la reducción de ruido se produce, aunque aparece un emborronado de la imagen. Hay que tener en cuenta que después la regresión sólo se aplicará sobre píxeles ruidosos. Aún así la diferencia con el original es mínima salvo en regiones homogéneamente blancas como el caso del cuello de la camisa en la imagen Albert. Este último efecto es debido a que la aplicación de la regresión se ha realizado de manera recursiva, de forma que los valores ya aproximados han sido utilizados para obtener los valores siguientes.

**Parámetros de entrenamiento.** Ya que los resultados se adecuan a lo esperado es necesario buscar los mejores valores y elegir el kernel que mejor se adapte al problema. En el caso del kernel se probaron varias opciones incluyendo el kernel gaussiano, ERBF, $\chi^2$ y splines. Los resultados obtenidos con kernel spline o $\chi^2$ están lejos de lo esperado y no son adecuados para la reducción de ruido, a pesar de su uso en otras aplicaciones de regresión. Sin embargo, los kernel gaussiano y ERBF sí que dan buenos resultados.

Una vez elegidos los kernel pasamos a la elección de los parámetros de entrenamiento. Debemos elegir $C$, $\gamma$ y $\varepsilon$. La constante $C$ influye en los resultados además de en el número de vectores soporte, aunque la mejora obtenida no es indefinida y si su valor crece más allá de 1000 las mejoras no son importantes o incluso se empeoran los resultados. En la figura 5.22 se muestra la variación del error cometido en la reconstrucción para distintos valores



**(a)** **(b)**

**(c)** **(d)**

**(e)** **(f)**

Figura 5.21: Ejemplos del proceso de regresión. (a), (b) Imagen con un 20 % de ruido impulsivo blanco; (c), (d) Imagen reconstruida; (e), (f) Diferencia con el original.

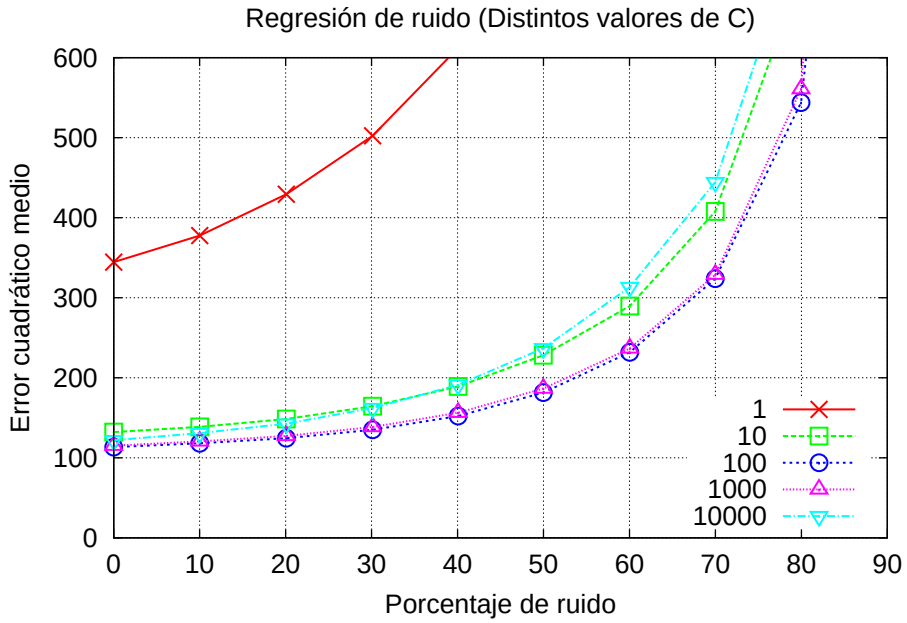


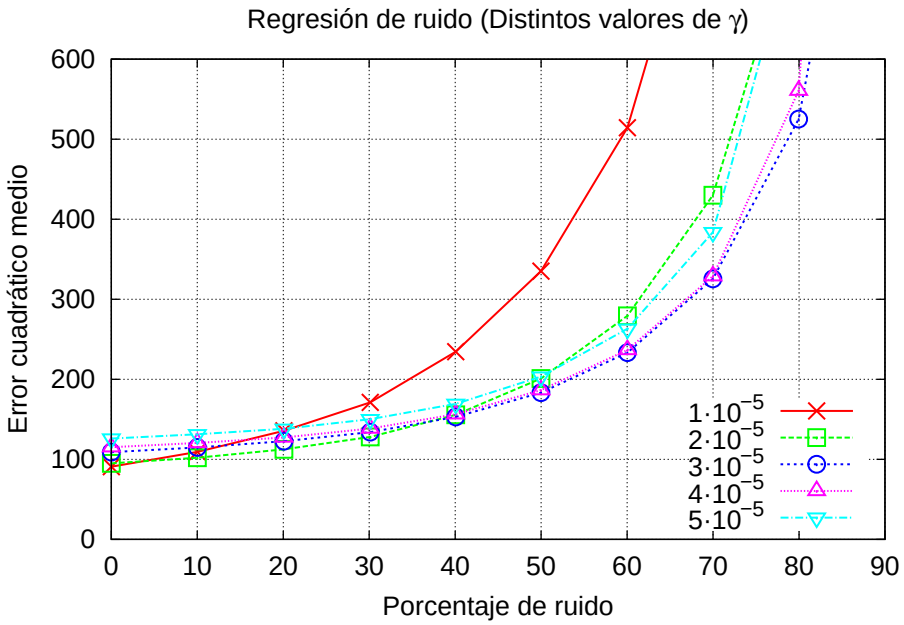

Figura 5.22: Resultados obtenidos en la reconstrucción de la imagen Lenna variando $C$. Se usó un kernel gaussiano con $\gamma = 4 \cdot 10^{-5}$.

de C, en esta prueba se utilizó el kernel gaussiano con una $\gamma = 4 \cdot 10^{-5}$. Puede comprobarse que los resultados mejoran si se aumenta $C$ pero llegado al valor $C = 1000$ no existe tal mejora. El mejor valor sería $C = 100$ pero, puesto que la mejora no es significativa se prefirió el uso de $C = 1000$ por la reducción de vectores soporte que supone, alrededor de 150 menos.

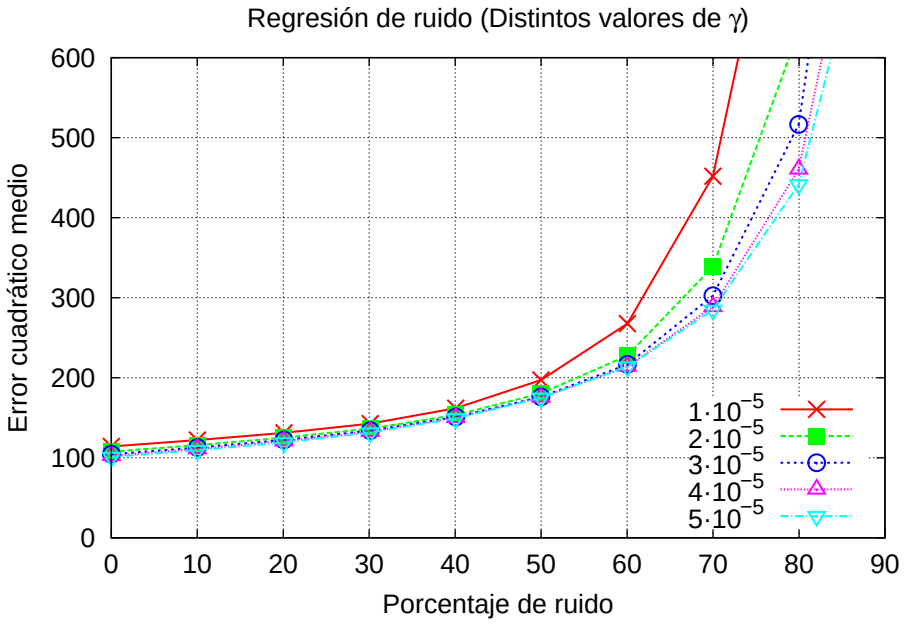
Para el ajuste de $\gamma$ se hizo un barrido con distintos valores y se usaron para reconstruir la imagen Lenna usando tanto el kernel gaussiano como el ERBF. En la figura 5.23 se muestran gráficamente los resultados obtenidos. En el caso del kernel gaussiano, el aumento de $\gamma$ produce mejores resultados hasta llegar a $\gamma = 5 \cdot 10^{-5}$ donde vuelve a empeorar. Por tanto, la elección estará entre $\gamma = 3 \cdot 10^{-5}$ y $\gamma = 4 \cdot 10^{-5}$. Para el kernel ERBF la mejora con $\gamma$ es continua pero al llegar a $\gamma = 5 \cdot 10^{-5}$ la diferencia con el valor anterior es mínima y, por tanto, esa será nuestra elección. Se comprueba que los resultados usando ERBF son mejores que con el kernel gaussiano aunque a costa de manejar más vectores soporte y, por tanto, reducir la velocidad.

Falta elegir el mejor valor de $\varepsilon$, que en las pruebas anteriores ha sido fijado a $\varepsilon = 0.001$. Se varió $\varepsilon$ en una forma similar a las pruebas anteriores pero no se constató una diferencia apreciable en reconstrucción o en número de vectores soporte. Sin embargo, el tiempo de entrenamiento se veía drásticamente aumentado para valores pequeños de $\varepsilon$. Ante estos resultados se optó por mantener el valor por defecto de $\varepsilon = 0.001$ porque el entrenamiento es relativamente rápido y el número de vectores era menor que para valores mayores de $\varepsilon$. Siempre teniendo en cuenta que la reconstrucción no se ve afectada.

**Elección del tamaño y del ruido de las imágenes de entrenamiento.** Al igual que en el caso de la detección de ruido, la elección del tamaño y del ruido de las imágenes




**(a)**




**(b)**

Figura 5.23: Variación del error cuadrático medio en la recuperación de la imagen en función de $\gamma$. (a) Kernel Gaussiano. (b) Kernel ERBF.

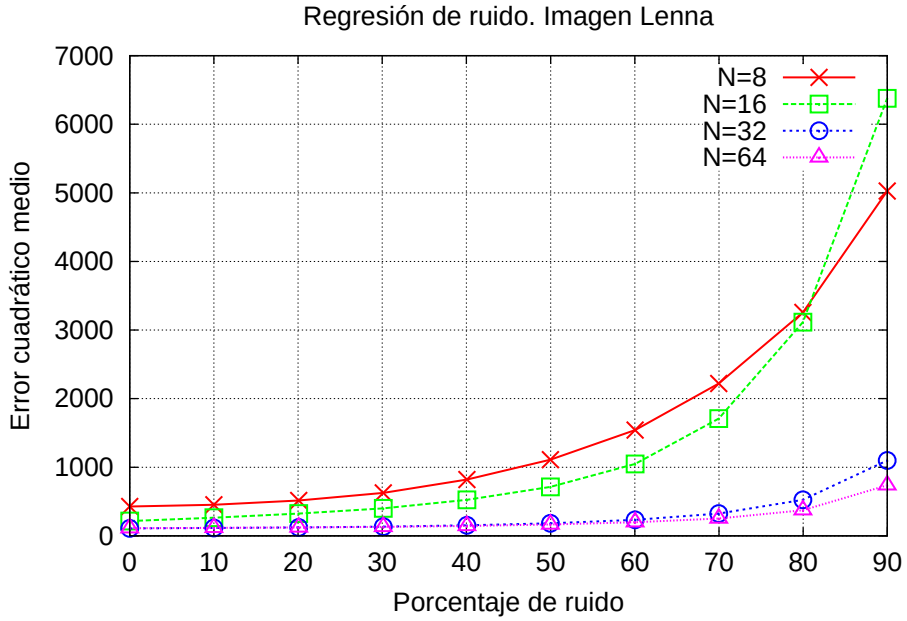



Figura 5.24: Resultados de regresión para la imagen Lenna usando distintos tamaños de entrenamiento. El ruido añadido es del 30 %.

de entrenamiento es importante para obtener los mejores resultados. El aumento del tamaño incrementa el número de vectores de entrenamiento, así como, el uso de una mayor variedad de niveles de gris con lo que se espera una mejora en la regresión con el aumento del mismo. Este aumento producirá también un mayor número de vectores soporte y, por tanto, una reducción de velocidad. El ruido añadido hará que el entrenamiento sintético se parezca al ruido existente en las imágenes reales aunque, a priori, es difícil decir cual es el nivel de ruido ideal.

Para la obtención de los mejores tamaños y ruidos para las imágenes de entrenamiento se optó por realizar distintas pruebas en las que se variaba sólo el tamaño o sólo el ruido añadido. En la figura 5.24 se presentan de manera gráfica los resultados obtenidos con la imagen Lenna y distintos tamaños de entrenamiento habiendo fijado el ruido de entrenamiento a un 30 %. Puede observarse como, efectivamente, un aumento de tamaño produce una mejora en los resultados pero a costa de un mucho mayor tiempo de entrenamiento y un mayor número de vectores soporte. Por ejemplo, de los 1192 vectores soporte obtenidos con un tamaño de 32 se pasa a 3184 con 64. Sin embargo, la mejora de calidad al pasar de 32 a 64 no es tan grande (sólo es apreciable para ruidos altos) como para preferir el tamaño mayor puesto que el tiempo de ejecución es prácticamente el doble. De hecho no se presentan datos para el tamaño 128 porque el tiempo de ejecución es tan grande que se descartó de entrada.

En la figura 5.25 se muestra el resultado de aplicar la regresión a la imagen Lenna para distintos ruidos de entrenamiento y con un tamaño de imagen de 32. En este caso se puede observar como el aumento del ruido de entrenamiento mejora los resultados hasta que al superar el 40 % estos empeoran. La conclusión a la que se puede llegar es que el ruido ideal para el entrenamiento debería situarse entre el 30 % y el 40 % sin un ganador claro puesto que aunque los resultados para ruido bajo son mejores para 40 % en ruido

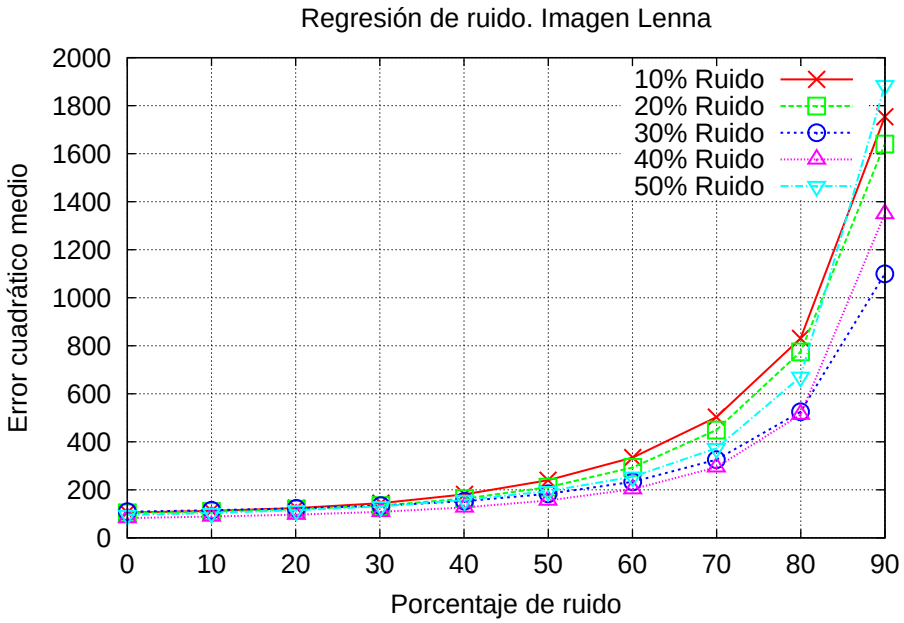



Figura 5.25: Resultados para distintos niveles de ruido de entrenamiento. El tamaño es fijo de 32.

alto no se mantiene esa distancia, además la diferencia no es excesivamente grande. Por otro lado, el aumento de vectores soporte de usar un 30 % a un 40 % no es excesivo, pasa de 1192 a 1396.

Estas pruebas dan una idea de la evolución de los resultados en función de las imágenes de entrenamiento empleadas y pueden usarse a la hora de aplicar los métodos de reducción de ruido que utilicen la regresión. Siempre teniendo en cuenta que este es un método costoso y que a veces habrá que sacrificar calidad a costa de mayor rapidez.

### 5.2.3. Filtros de mediana modificados usando SVM o RBF

Uno de los métodos clásicos de reducción de ruido impulsivo es el filtro de mediana (Sección 5.1.2) y además es la base de otro buen número de métodos que utilizan sus propiedades. El principal problema del filtro de mediana como se ha visto, es el hecho de que se aplica por igual a todos los píxeles de la imagen independientemente de que estos sean ruidosos o no. Esto se puede evitar utilizando el esquema de clasificación presentado en la figura 5.1 para detectar previamente los píxeles ruidosos antes de su sustitución. Por tanto, la primera solución propuesta parte de la definición de un clasificador eficiente de píxeles ruidosos usando las ideas ya presentadas en otros capítulos como son el uso de imágenes sintéticas de entrenamiento junto con SVM o alternativas como las RBF. Por tanto, en este método la reconstrucción se basará en la calidad del clasificador. Una versión inicial del proceso presentado en este punto se publicó en [Gómez-Moreno01c].

En este caso, por tanto, el filtro de mediana se utilizará como herramienta para la reconstrucción de los píxeles detectados previamente como ruidosos. Esta idea general da lugar a distintas posibilidades que se resumen a continuación:

1.- Aplicar directamente la mediana en el entorno del píxel ruidoso incluyéndolo a él mismo. Esta posibilidad es muy similar a la expuesta a continuación y las pruebas iniciales realizadas producían peores resultados.

2.- Aplicar la mediana sobre el entorno pero excluyendo el píxel que ya se sabe ruidoso al igual que se realiza en [Abreu96] y donde se llama a esta técnica ROM (Rank Ordered Mean). Llamaremos a este método **SVM-Mediana-1**.

3.- Puesto que podemos usar el clasificador previo para marcar todos los píxeles ruidosos esta información se puede usar para utilizar en la mediana sólo los píxeles no ruidosos. Este otro método es el **SVM-Mediana-2**.

A esto hay que añadir el que se pueden aplicar de manera recursiva o no recursiva mediante ciclos repetitivos.

Estas posibilidades cuentan con ventajas e inconvenientes como se mostrará a continuación. La primera técnica tiene a su favor la simplicidad y en contra el que el número de píxeles ruidosos que permanecen en la imagen puede ser grande para ruidos altos. La segunda opción supone una mejora sobre la anterior pero sigue teniendo los mismos problemas en cuanto a la no eliminación de píxeles. La última opción es la más compleja en cuanto a implementación aunque supone una mejora importante en la recuperación de los píxeles. Sólo para porcentajes de ruido muy altos el valor de reconstrucción seguirá siendo ruido porque todo el entorno sea ruidoso. En ese caso una aplicación no recursiva mejorará los resultados.

**Mediana modificada usando el método SVM-Mediana-1.** En la figura 5.26 se presentan algunos ejemplos obtenidos sobre la imagen Lenna que permiten observar las diferencias entre la aplicación recursiva y no recursiva del método SVM-Mediana-1. En las figuras 5.26(a) y 5.26(b) aparecen las imágenes resultantes de la aplicación de la mediana $3 \times 3$ sin ninguna modificación sobre la imagen con un 20 % y un 60 % de ruido añadido. Puede verse que el problema de la mediana es que muchos píxeles ruidosos permanecen en la imagen después de su aplicación.

En las figuras 5.26(c) y 5.26(d) tenemos dos ejemplos de la aplicación del filtro de mediana modificado usando el método SVM-Mediana-1 y de manera recursiva. Esta forma de aplicación tiene la ventaja de la rápidez pues no necesita de su repetición para limpiar la imagen. Para ruidos bajos como el caso del 20 % presentado en 5.26(c) los resultados son buenos y se consigue la eliminación efectiva del ruido. Sin embargo, para ruidos más altos como el 60 % presente en 5.26(d) se produce un efecto de arrastre por el cual ciertas zonas de ruido que no han podido eliminarse completamente "manchan" otras zonas adyacentes en un efecto de arrastre. Este es un problema común a la recursividad. En los dos casos se han usado imágenes de entrenamiento de $32 \times 32$ con un 40 % de ruido.

Para evitar ese efecto se puede realizar la aplicación de manera no recursiva. Este tipo de aplicación tiene el problema de que pueden quedar píxeles ruidosos aislados si el ruido es grande. Para evitar ese problema se ha de repetir la aplicación del filtro sobre la imagen las veces que se considere necesario hasta la eliminación del ruido. En los ejemplos mostrados en las figuras 5.26(e) y 5.26(f) se ha aplicado el filtro hasta que el porcentaje de ruido detectado cae por debajo del 0.1 %. Esto implica que la eliminación es rápida para un ruido pequeño y más lenta cuando el ruido crece, por ello, se ha puesto un límite

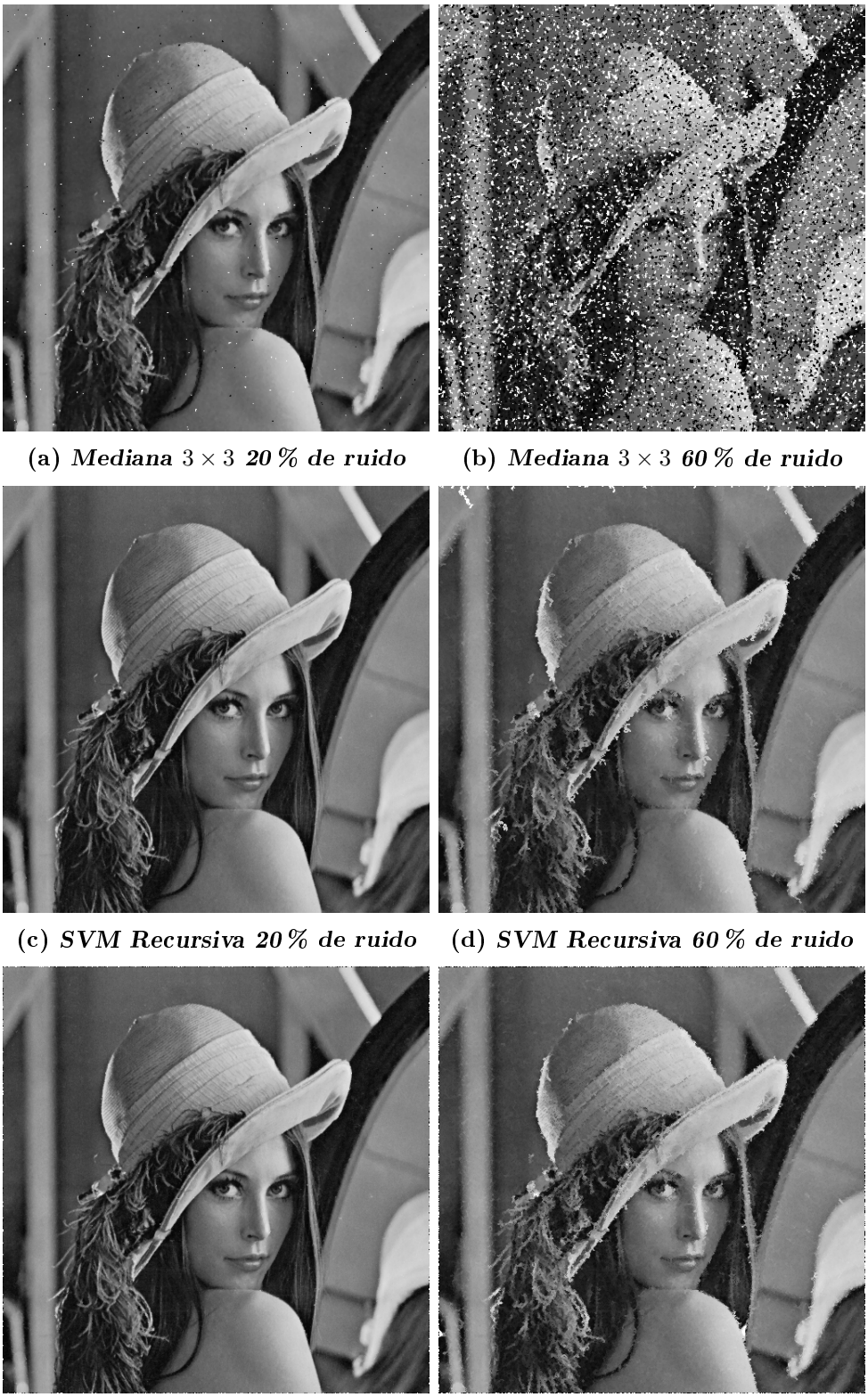
de 25 repeticiones para no hacer excesivamente largo el proceso. Puede comprobarse como el resultado para ruido de 20 % es muy similar y con el 60 % el efecto de arrastrado se reduce. El entrenamiento usado es el mismo que el de la aplicación recursiva.

Los resultados de detección del ruido obtenidos con las SVM son muy buenos como se vio en la sección 5.2.1 y se puede apreciar en los ejemplos presentados en la figura 5.26. Sin embargo, no es la única posibilidad de clasificación eficiente. Como comparación y también para explorar otras posibilidades se ensayó el uso de redes RBF en la modificación del filtro de mediana.

En este caso para el entrenamiento se usó las mismas imágenes sintéticas ya usadas para las SVM. En cuanto a la elección de parámetros, en este caso sólo hay que fijar a priori el número de neuronas de la capa oculta y, partiendo de un número inicial de clusters, realizar el cálculo de los centroides y los pesos de cada neurona. El número de neuronas se ha elegido haciendo que sea similar al número de vectores soporte obtenidos tras el mejor entrenamiento, que para imágenes de $32 \times 32$ y un 40 % de ruido es de 50 neuronas.

Con las RBF el problema que se ha encontrado al realizar el entrenamiento es que no siempre se obtiene el mejor modelo posible sino que, dependiendo del punto de partida (aleatorio), los resultados pueden ser muy diferentes. Por tanto, es necesario repetirlo varias veces y comprobar sus resultados de detección antes de dar por bueno un modelo. Esa es la única diferencia entre el uso de las SVM o de las redes RBF. En la figura 5.27 se muestran algunos ejemplos de la aplicación de la mediana modificada usando RBF tanto de manera recursiva como no recursiva. Sus imágenes se pueden comparar con las de la figura 5.26 y se verá que la principal diferencia es que siguen apareciendo píxeles ruidosos (especialmente el ruido negro) tanto para nivel bajos como altos de ruido. Esto implica que la clasificación no es del todo eficiente aunque en las pruebas de entrenamiento realizadas no se consiguió un resultado mejor.

**Mediana modificada usando el método SVM-Mediana-2.** En el apartado anterior se ha comprobado que el uso de un clasificador junto con un filtro de mediana en el que el píxel detectado como ruido no se tiene en cuenta mejora los resultados obtenidos con el tradicional filtro de mediana. Sin embargo ese método no utiliza del todo las posibilidades ofrecidas por la clasificación. Ya que podemos clasificar los píxeles de la imagen entre ruidosos y no ruidosos, parece conveniente que para la reconstrucción sólo se usen aquellos no ruidosos. Por ello en el método SVM-Mediana-2 primero se procede a la detección de los píxeles ruidosos y se marcan para, posteriormente, realizar la reconstrucción de dichos píxeles usando la mediana. Sin embargo, en esta mediana sólo se usan los píxeles no ruidosos y, puesto que el número de píxeles cambia en cada posición, la elección del valor de reconstrucción será aquel que ocupe la posición $\frac{L-1}{2}$ donde $L$ representa el número de píxeles usados para la obtención de la mediana. Por supuesto, puede ocurrir que si el ruido es grande no haya píxeles no ruidosos alrededor del píxel a reconstruir, en ese caso no se obtiene valor de reconstrucción y el píxel se queda igual. Esto se puede evitar de dos maneras, o mediante la aplicación recursiva del filtro o mediante la repetición de su aplicación. Se ha optado por la aplicación repetitiva para evitar el efecto de arrastrado propio de la recursividad y, de esa manera, mejorar los resultados visuales. Todo ello a costa de un mayor tiempo de ejecución cuando el ruido es muy grande.



(a) ***Mediana*** $3 \times 3$ ***20 % de ruido*** (b) ***Mediana*** $3 \times 3$ ***60 % de ruido***

(c) ***SVM Recursiva 20 % de ruido*** (d) ***SVM Recursiva 60 % de ruido***

(e) ***SVM No recursiva 20 % de ruido*** (f) ***SVM No recursiva 60 % de ruido***

Figura 5.26: Aplicación de la técnica de mediana modificada usando SVM (SVM-Mediana-1). Se incluye en (a) y (b) los resultados para la mediana clásica.

(a) ***RBF recursiva 20 % de ruido*** (b) ***RBF recursiva 60 % de ruido***

(c) ***RBF no recursiva 20 % de ruido*** (d) ***RBF no recursiva 60 % de ruido***

Figura 5.27: Aplicación de la técnica de mediana modificada con RBF.

(a) ***SVM-Mediana-2 20 % de ruido***

(b) ***SVM-Mediana-2 60 % de ruido***

(c) ***SVM-Mediana-2 90 % de ruido***

Figura 5.28: Aplicación de la técnica de mediana modificada con SVM en su segunda versión (SVM-Mediana-2).

En la figura 5.28 tenemos un ejemplo visual de la aplicación de esta técnica. Se puede apreciar que incluso con ruidos muy altos (90 %) el resultado es aceptable. No aparece el efecto de arrastrado y tampoco quedan píxeles ruidosos aislados.

### 5.2.4. Filtros SVM con regresión

Un paso más en la aplicación de las SVM a la reconstrucción de imágenes ruidosas es la inclusión de la regresión en el proceso de recuperación de valores. Como se vio en el apartado 5.2.2 la regresión con SVM se puede usar en la eliminación de ruido impulsivo aunque su sola aplicación produce un efecto de emborronado que también es propio de otras técnicas como la mediana. La propuesta en este apartado es su aplicación como parte de un sistema de clasificación y reconstrucción (Figura 5.1). La regresión haría el trabajo que realizaba la mediana en el punto anterior, siempre a costa de una mayor carga computacional en cada píxel reconstruido. Sin embargo, se espera que la obtención de valores mediante regresión permita no tener que aplicar el método de manera repetitiva y poder hacerlo de manera recursiva. Esto haría que el tiempo de ejecución se redujera drásticamente.

La figura 5.29 permite apreciar las diferencias de resultados con respecto a las técnicas de mediana modificada. Para valores pequeños y medianos de ruido los resultados son comparables y, a simple vista, sólo se aprecian diferencias en las zonas correspondientes a los bordes de los objetos aunque, en general, la reconstrucción es buena. Sin embargo, para ruidos muy altos el efecto de arrastrado acaba por aparecer y aunque se reconstruye la imagen y se puede apreciar qué objetos la componen, el resultado es inferior al de la técnica SVM-Mediana-2. En todo caso hay que tener en cuenta que la aplicación no recursiva realizada con SVM-Mediana-2 implica un filtrado repetido 25 veces mientras que la regresión sólo se aplica una vez.

## 5.3. Evaluación y presentación de resultados

Llegados a este punto sólo se han ofrecido algunos ejemplos de resultados de los métodos diseñados en esta tesis de manera visual. Sin embargo no se ha realizado una comparación entre ellos ni con otros métodos de la literatura. Para realizar dicha comparación primero se presentarán algunas medidas de calidad usadas en reconstrucción de imágenes y se decidirá cuales de ellas serán finalmente usadas. Posteriormente se compararán los distintos métodos propuestos, entre sí y con los de la literatura. Esta comparación se realizará para varias imágenes, pues ya se ha comprobado que los resultados dependen del tipo de imagen usado, de la presencia de texturas o de zonas uniformes en las mismas.

### 5.3.1. Medidas de calidad

En el ámbito de la reconstrucción de imágenes se han definido varias medidas de calidad para saber si el método usado es más o menos eficiente. Algunas de ellas se definen a continuación:

(a) ***SVM regresión 20 % de ruido***

(b) ***SVM regresión 60 % de ruido***

(c) ***SVM regresión 90 % de ruido***

Figura 5.29: Aplicación de la técnica de SVM usando además la regresión.

1.- **Mean Squared Error** ($MSE$). Error cuadrático medio cometido en la reconstrucción. Se define según la siguiente expresión:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}||I(x,y) - \hat{I}(x,y)||^2,$$

donde $m \times n$ son las dimensiones de la imagen. Se supone que cuanto más pequeña es esta medida mejor es la calidad de reconstrucción.

2.- **Peak Signal to Noise Ratio** ($PSNR$). Esta está relacionada con la medida anterior. Es una medida logarítmica y se utiliza ampliamente en procesado de imagen. Se define según la siguiente expresión:

$$PSNR(dB) = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right),$$

en imágenes de escala de grises con 8 bits por pixel $MAX_I = 255$. Con esta medida, un valor mayor implica una mejor calidad. Valores de $PSNR$ por encima de 40 dB implican una reconstrucción casi perfecta.

3.- **Mean Absolute Error** ($MAE$). Es el error absoluto medio que se utiliza en estadística para medir la distancia entre una predicción y la realidad. En el caso de imágenes se define como:

$$MAE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}||I(x,y) - \hat{I}(x,y)||,$$

y al igual que el $MSE$ un valor menor implica una mejor calidad.

4.- **Structural SIMilarity** ($SSIM$). Esta es una medida de calidad en la que no sólo se mide la diferencia entre las dos imágenes sino que se miden una serie de parámetros estructurales para comprobar la similitud. Esos parámetros estructurales son medidos de manera independiente de los valores de gris o la iluminación de la imagen. Una descripción detallada de esta medida puede verse en [Wang04].

La expresión que define la similitud entre dos ventanas, del mismo tamaño, de dos imágenes distintas viene dada por la siguiente expresión:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{5.7}$$

dónde $\mu_x$ y $\mu_y$ son las medias de $x$ e $y$, $\sigma_{xy}$ es la covarianza entre $x$ e $y$, $\sigma_x^2$ y $\sigma_y^2$ son las varianzas de $x$ e $y$ y las constantes $c_1$ y $c_2$ permiten estabilizar la división en el caso de que las medias o las varianzas sean próximas a cero.

Con dicha expresión se obtendría un mapa de valores de $SSIM$ correspondientes a bloques de las imágenes comparadas. Como realmente se quiere una medida global de calidad el coeficiente que se utiliza para comparar dos imágenes completas es el $MSSIM$ (Mean SSIM) que se define:

$$\mathrm{MSSIM}(X,Y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j, y_j), \tag{5.8}$$

donde $X$ e $Y$ son las imágenes a comparar, $M$ es el numero de bloques en los que se dividen las imágenes y $x_j$ e $y_j$ son los bloques a comparar.

Un valor cercano a 1 en esta medida indica una buena calidad mientras que conforme la calidad empeora la medida tiende a 0. La principal ventaja de este índice de calidad es que no da tanta importancia a los errores puntuales de reconstrucción y sí a la calidad estructural en su conjunto, además de tener una relación directa con la apreciación subjetiva de calidad.

Tras unas pruebas iniciales y la consulta de la literatura al respecto se decidió el uso del PSNR pues es la más extendida y favorece la comparación con otros métodos y del MSSIM porque es el que mejor relación tiene con los aspectos psicovisuales que indican la calidad de reconstrucción de una imagen. Tanto el MSE como el MAE están relacionadas con el PSNR por lo que no aportan más información para la comparación deseada.

### 5.3.2. Imágenes utilizadas

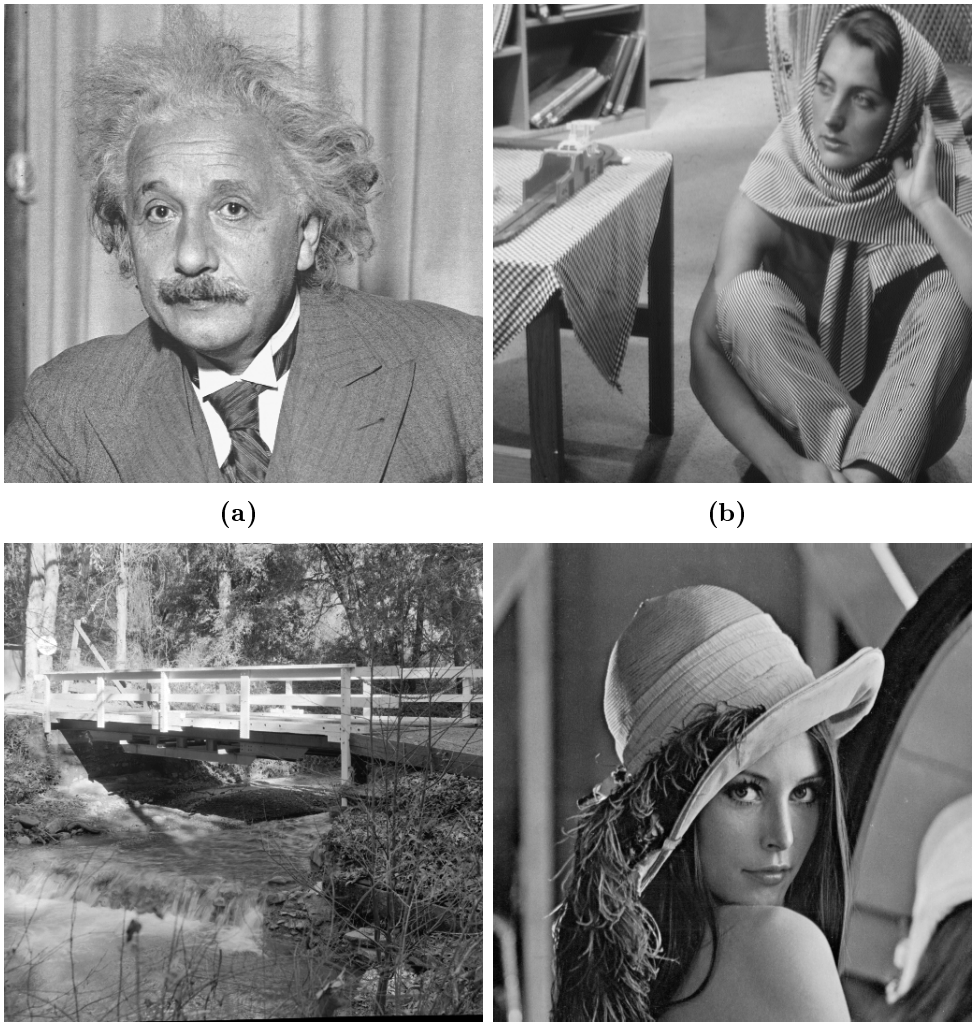
En la realización de los tests para comparar los distintos métodos propuestos y los seleccionados de la literatura se han elegido 4 imágenes basándonos en las características que les hacen diferentes, en su presencia en distintos trabajos de reducción de ruido ([Lightstone95, Abreu96, Russo00, Schulte06a]) y en el grado de desafío que suponen.

En la figura 5.30 se muestran dichas imágenes elegidas y se pueden apreciar las características que se enuncian a continuación (algunas de ellas ya se discutieron en el apartado 5.2.1):

- **Albert**. Está imagen resulta desafiante por la parte texturada del traje pero sobre todo por el cuello blanco de la camisa donde la detección del ruido se hace más difícil. Sobre todo en los bordes como ya se comentó anteriormente.
- **Barbara.** Está es una imagen muy desafiante debido a la gran cantidad de texturas y patrones que se repiten en la misma. La reconstrucción de la rayas resulta muy difícil para todos los algoritmos de reducción de ruido.
- **Bridge.** Esta imagen también presenta algunos problemas para la detección del ruido en dos zonas, una blanca próxima al puente y el puente mismo y una negra producto de un mal escaneado. Estas zonas uniformes próximas a los extremos de la escala de grises son especialmente difíciles de tratar.
- **Lenna.** Una imagen muy utilizada en comparación de algoritmos de eliminación de ruido aunque no presenta grandes problemas. Tienen zonas uniformes de gris pero no hay blancos o negros. Sirve como contrapunto para comparar respecto a las anteriores.

### 5.3.3. Evaluación y comparación

Una vez decididas las medidas de evaluación y las imágenes sobre las que probar se procederá a la obtención de datos y su presentación para la mejor comparación de métodos. En todos los casos se presentarán los datos tanto de forma numérica, para tener



**(a)** **(b)**

**(c)** **(d)**

Figura 5.30: Imágenes utilizadas en los tests de los distintos métodos de eliminación de ruido. (a) Albert, (b) Barbara, (c) Bridge, (d) Lenna

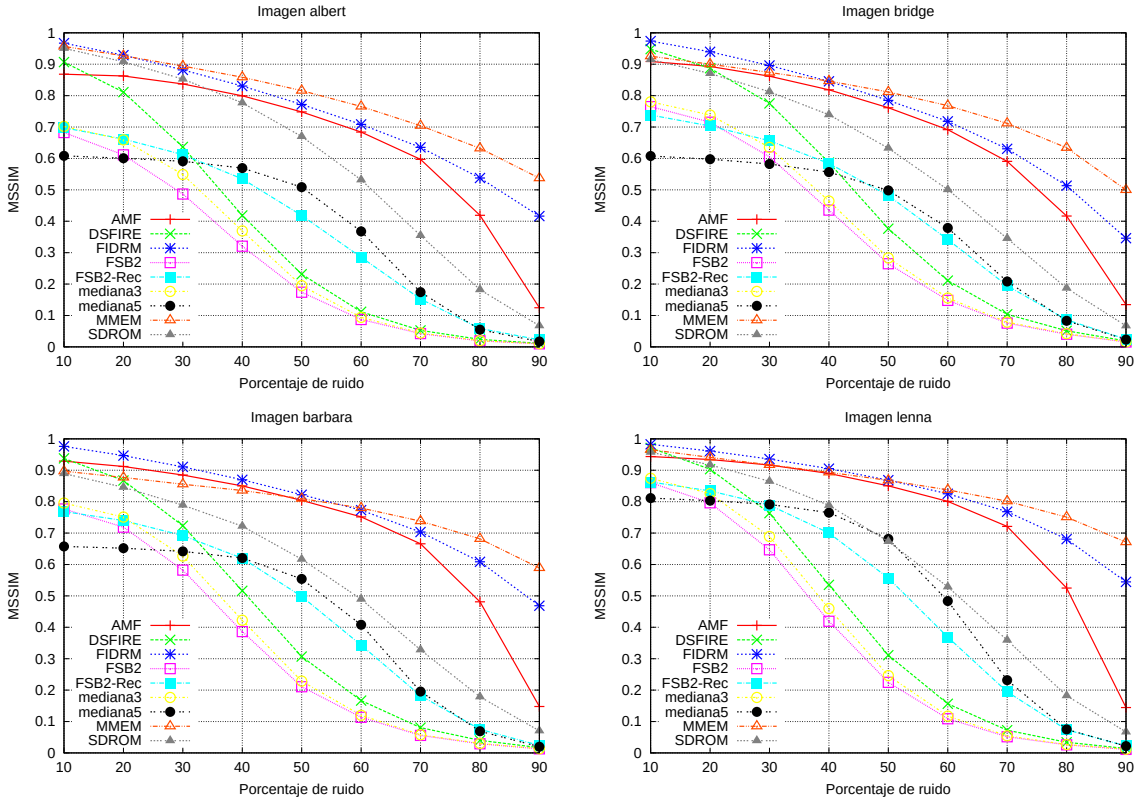



Figura 5.31: Comparación gráfica del parámetro MSSIM para los distintos métodos de la literatura.

acceso a toda la información, como de forma gráfica para poder realizar las comparaciones de una manera visual.

**Comparación de métodos en la literatura.** El primer paso en esta sección será la comparación de algunos de los métodos presentados en la literatura respecto a la eliminación de ruido impulsivo. Se han elegido el clásico de mediana con dos longitudes de ventana, el SDROM y el AMF (Adaptative Median Filter) como filtros de mediana modificados, el MMEM como método diseñado especialmente para la eliminación de ruido "Salt&Pepper" y algunos basados en lógica difusa (FIDRM, DSFIRE, FSB).

Del estudio de las tablas 5.1 y 5.2 y de las figuras 5.31 y 5.32 se puede llegar a las siguientes conclusiones:

1.- Ninguno de los métodos estudiados en la literatura presenta un mejor comportamiento en todos los escenarios, es decir, para todos los ruidos añadidos o para todas las imágenes.

2.- Existe relación entre el índice PSNR y el MSSIM en cuanto a la calidad de reconstrucción (Un mal PSNR supone un mal MSSIM pero el mejor PSNR no es el mejor MSSIM) aunque hay ligeras variaciones. Aún así, ambas aportan información para la comprobación de la calidad.

3.- Los métodos que destacan son el MMEM por sus buenos resultados, sobre todo para niveles altos de ruido. El FIDRM que es el mejor de los basados en lógica difusa y

Tabla 5.1: Tablas de resultado MSSIM para los distintos métodos de la literatura.

| Imagen albert | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | AMF | DSFIRE | FIDRM | FSB2 | FSB2-Rec | mediana3 | mediana5 | MMEM | SDROM |
| 10 | 0.868121 | 0.906634 | 0.967178 | 0.682366 | 0.697717 | 0.702015 | 0.608270 | 0.956672 | 0.950471 |
| 20 | 0.862645 | 0.811188 | 0.928897 | 0.611336 | 0.661726 | 0.658993 | 0.600737 | 0.926977 | 0.907875 |
| 30 | 0.837262 | 0.636512 | 0.881967 | 0.486744 | 0.612256 | 0.548848 | 0.590896 | 0.894313 | 0.852711 |
| 40 | 0.799501 | 0.418459 | 0.830517 | 0.320572 | 0.534943 | 0.369357 | 0.569344 | 0.858486 | 0.776943 |
| 50 | 0.748164 | 0.231303 | 0.771919 | 0.173990 | 0.418555 | 0.194878 | 0.508645 | 0.815968 | 0.669926 |
| 60 | 0.683942 | 0.111906 | 0.708554 | 0.087091 | 0.285450 | 0.094001 | 0.367654 | 0.765509 | 0.531835 |
| 70 | 0.596994 | 0.052378 | 0.635423 | 0.042420 | 0.152435 | 0.044076 | 0.174375 | 0.704465 | 0.354679 |
| 80 | 0.419037 | 0.023863 | 0.537767 | 0.019216 | 0.058674 | 0.019990 | 0.054745 | 0.632327 | 0.182095 |
| 90 | 0.124649 | 0.011254 | 0.416417 | 0.009609 | 0.022519 | 0.009829 | 0.017113 | 0.537140 | 0.067399 |

| Imagen bridge | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | AMF | DSFIRE | FIDRM | FSB2 | FSB2-Rec | mediana3 | mediana5 | MMEM | SDROM |
| 10 | 0.908810 | 0.947906 | 0.973761 | 0.765018 | 0.738382 | 0.780388 | 0.607672 | 0.925546 | 0.914898 |
| 20 | 0.892997 | 0.887390 | 0.939651 | 0.715296 | 0.703237 | 0.738277 | 0.597509 | 0.900200 | 0.870606 |
| 30 | 0.861833 | 0.775278 | 0.896047 | 0.605348 | 0.656542 | 0.635272 | 0.582359 | 0.873745 | 0.812396 |
| 40 | 0.818482 | 0.586901 | 0.845570 | 0.435419 | 0.584055 | 0.464307 | 0.556484 | 0.846285 | 0.739280 |
| 50 | 0.761476 | 0.376304 | 0.785036 | 0.264658 | 0.481817 | 0.283151 | 0.498125 | 0.811575 | 0.632748 |
| 60 | 0.691620 | 0.211070 | 0.718108 | 0.148161 | 0.341788 | 0.155737 | 0.378589 | 0.768116 | 0.500707 |
| 70 | 0.590708 | 0.103841 | 0.630632 | 0.075724 | 0.194878 | 0.078813 | 0.208312 | 0.711744 | 0.345197 |
| 80 | 0.416639 | 0.051343 | 0.513291 | 0.040348 | 0.086126 | 0.042046 | 0.083188 | 0.633920 | 0.187178 |
| 90 | 0.134509 | 0.018906 | 0.346012 | 0.015432 | 0.024463 | 0.015953 | 0.023104 | 0.499710 | 0.067004 |

| Imagen barbara | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | AMF | DSFIRE | FIDRM | FSB2 | FSB2-Rec | mediana3 | mediana5 | MMEM | SDROM |
| 10 | 0.929117 | 0.937931 | 0.975972 | 0.776707 | 0.768589 | 0.795651 | 0.657475 | 0.896966 | 0.889462 |
| 20 | 0.911970 | 0.867551 | 0.946770 | 0.718946 | 0.739782 | 0.750945 | 0.651946 | 0.877041 | 0.845796 |
| 30 | 0.884692 | 0.723235 | 0.910910 | 0.582008 | 0.692155 | 0.626701 | 0.641808 | 0.855632 | 0.789102 |
| 40 | 0.850235 | 0.516231 | 0.869650 | 0.386429 | 0.619710 | 0.422405 | 0.620539 | 0.835505 | 0.722051 |
| 50 | 0.804300 | 0.306824 | 0.822102 | 0.210690 | 0.498192 | 0.228621 | 0.553983 | 0.809643 | 0.616627 |
| 60 | 0.751471 | 0.166767 | 0.772958 | 0.113381 | 0.343290 | 0.119821 | 0.407984 | 0.779711 | 0.489460 |
| 70 | 0.665855 | 0.080529 | 0.703291 | 0.055785 | 0.182105 | 0.057745 | 0.195380 | 0.738071 | 0.328331 |
| 80 | 0.481255 | 0.040024 | 0.608531 | 0.029157 | 0.075259 | 0.030282 | 0.069086 | 0.681819 | 0.178648 |
| 90 | 0.147859 | 0.016713 | 0.468699 | 0.012874 | 0.023842 | 0.013352 | 0.019730 | 0.588916 | 0.070370 |

| Imagen lenna | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | AMF | DSFIRE | FIDRM | FSB2 | FSB2-Rec | mediana3 | mediana5 | MMEM | SDROM |
| 10 | 0.0,943772 | 0.0,971202 | 0.0,982776 | 0.0,860725 | 0.0,862628 | 0.0,874411 | 0.0,811452 | 0.0,965510 | 0.0,957541 |
| 20 | 0.0,934364 | 0.0,903364 | 0.0,961310 | 0.0,796288 | 0.0,834273 | 0.0,825765 | 0.0,803241 | 0.0,940751 | 0.0,917074 |
| 30 | 0.0,916601 | 0.0,763631 | 0.0,935665 | 0.0,646693 | 0.0,788880 | 0.0,688976 | 0.0,791170 | 0.0,917256 | 0.0,864853 |
| 40 | 0.0,889887 | 0.0,535094 | 0.0,905280 | 0.0,419329 | 0.0,700295 | 0.0,459854 | 0.0,765031 | 0.0,893362 | 0.0,788234 |
| 50 | 0.0,850511 | 0.0,311073 | 0.0,868523 | 0.0,224678 | 0.0,556897 | 0.0,246188 | 0.0,681422 | 0.0,867704 | 0.0,673200 |
| 60 | 0.0,801044 | 0.0,156423 | 0.0,825154 | 0.0,108717 | 0.0,368668 | 0.0,117269 | 0.0,483776 | 0.0,837583 | 0.0,528740 |
| 70 | 0.0,721114 | 0.0,072300 | 0.0,768082 | 0.0,051556 | 0.0,194603 | 0.0,054630 | 0.0,231585 | 0.0,801534 | 0.0,359055 |
| 80 | 0.0,524986 | 0.0,033892 | 0.0,680806 | 0.0,025728 | 0.0,073536 | 0.0,026869 | 0.0,074988 | 0.0,750824 | 0.0,182144 |
| 90 | 0.0,144542 | 0.0,013992 | 0.0,544621 | 0.0,011152 | 0.0,022141 | 0.0,011631 | 0.0,021683 | 0.0,671322 | 0.0,066667 |

Tabla 5.2: Tablas de resultado PSNR para los distintos métodos de la literatura.

| Imagen albert | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | AMF | DSFIRE | FIDRM | FSB2 | FSB2-Rec | mediana3 | mediana5 | MMEM | SDROM |
| 10 | 32.156223 | 33.030144 | 37.333382 | 26.003632 | 28.250486 | 29.054525 | 27.798508 | 33.922028 | 34.037266 |
| 20 | 31.494225 | 28.744181 | 33.819706 | 24.593628 | 27.322350 | 26.910271 | 27.463371 | 32.612499 | 31.648333 |
| 30 | 30.287420 | 23.970034 | 31.393667 | 21.699209 | 25.621080 | 23.136303 | 27.067673 | 31.541697 | 29.499716 |
| 40 | 28.988165 | 19.561769 | 29.766779 | 18.010174 | 23.237137 | 18.783873 | 26.149292 | 30.642998 | 27.370152 |
| 50 | 27.496731 | 16.023037 | 28.324076 | 14.840926 | 20.042292 | 15.282698 | 23.454302 | 29.655785 | 24.988792 |
| 60 | 25.917393 | 13.079757 | 27.036194 | 12.187391 | 16.905930 | 12.480180 | 19.029972 | 28.604557 | 22.217510 |
| 70 | 23.785048 | 10.586818 | 25.716215 | 9,907737 | 13.265514 | 10.087392 | 14.249015 | 27.454901 | 18.909363 |
| 80 | 19.116053 | 8,717817 | 24.014885 | 8,110012 | 9.931652 | 8,225535 | 10.481412 | 26.237661 | 15.080471 |
| 90 | 12.607323 | 7,199218 | 21.780247 | 6,678466 | 7,340016 | 6,736102 | 7,639777 | 24.485977 | 11.367532 |

| Imagen bridge | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | AMF | DSFIRE | FIDRM | FSB2 | FSB2-Rec | mediana3 | mediana5 | MMEM | SDROM |
| 10 | 28.918736 | 30.928211 | 33.963520 | 23.965105 | 25.230474 | 26.140736 | 23.983677 | 28.158722 | 28.426336 |
| 20 | 27.458090 | 27.638695 | 30.437441 | 22.949558 | 24.451977 | 24.687725 | 23.681313 | 27.326160 | 26.883114 |
| 30 | 25.876610 | 24.038748 | 28.022972 | 20.558146 | 23.285429 | 21.720373 | 23.264902 | 26.683672 | 25.150335 |
| 40 | 24.572765 | 19.934296 | 26.230507 | 17.357635 | 21.441824 | 18.055481 | 22.547857 | 26.050400 | 23.579012 |
| 50 | 23.170425 | 16.485783 | 24.720730 | 14.335366 | 19.048935 | 14.770107 | 20.834574 | 25.321407 | 21.657101 |
| 60 | 21.776451 | 13.432514 | 23.397732 | 11.688349 | 15.858322 | 11.980280 | 17.527256 | 24.499048 | 19.504391 |
| 70 | 20.127882 | 10.952327 | 22.064522 | 9,535521 | 12.721590 | 9,729467 | 13.553963 | 23.470270 | 16.967669 |
| 80 | 17.213848 | 8,990324 | 20.487640 | 7,790572 | 9,688658 | 7,908584 | 10.071818 | 22.280384 | 14.115790 |
| 90 | 11.722597 | 7,320960 | 18.422403 | 6,341210 | 6,861660 | 6,405240 | 7,233734 | 20.433199 | 10.699737 |

| Imagen barbara | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | AMF | DSFIRE | FIDRM | FSB2 | FSB2-Rec | mediana3 | mediana5 | MMEM | SDROM |
| 10 | 28.683268 | 29.068378 | 32.939953 | 23.099810 | 24.327742 | 24.761087 | 22.961891 | 26.428156 | 26.378366 |
| 20 | 27.489840 | 26.556208 | 29.426489 | 22.188246 | 23.701086 | 23.498623 | 22.782213 | 26.032326 | 25.491571 |
| 30 | 26.269314 | 23.302284 | 27.290861 | 20.046095 | 22.694765 | 20.987480 | 22.476858 | 25.635643 | 24.479036 |
| 40 | 25.104719 | 19.604736 | 25.680399 | 17.068670 | 21.146313 | 17.683439 | 21.942427 | 25.260748 | 23.404718 |
| 50 | 23.845617 | 16.171654 | 24.345676 | 14.121267 | 18.647657 | 14.507828 | 20.409649 | 24.752451 | 21.769831 |
| 60 | 22.729691 | 13.311886 | 23.339054 | 11.650817 | 15.856824 | 11.929062 | 17.546261 | 24.142450 | 19.995382 |
| 70 | 21.060757 | 10.877365 | 22.186180 | 9,526476 | 12.659291 | 9,707632 | 13.589187 | 23.317726 | 17.388323 |
| 80 | 17.855021 | 8,923391 | 20.894852 | 7,816155 | 9,619473 | 7,932400 | 10.117024 | 22.343952 | 14.370651 |
| 90 | 12.178355 | 7,353618 | 19.036133 | 6,418678 | 7,090191 | 6,478692 | 7,384542 | 20.923389 | 11.035494 |

| Imagen lenna | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | AMF | DSFIRE | FIDRM | FSB2 | FSB2-Rec | mediana3 | mediana5 | MMEM | SDROM |
| 10 | 36.202995 | 36.608616 | 39.575054 | 28.972931 | 30.242826 | 32.146908 | 29.419661 | 35.750389 | 34.718796 |
| 20 | 34.203274 | 31.090616 | 35.537968 | 26.624399 | 28.443504 | 28.486984 | 28.840775 | 33.860439 | 31.704111 |
| 30 | 32.254276 | 26.168858 | 33.024055 | 22.526293 | 26.124405 | 23.485781 | 28.118782 | 32.523937 | 29.379898 |
| 40 | 30.297773 | 21.193623 | 30.987045 | 18.155989 | 23.253441 | 18.715637 | 26.788357 | 31.411776 | 26.858957 |
| 50 | 28.139463 | 17.157440 | 29.266886 | 14.681046 | 19.778631 | 15.038507 | 23.445940 | 30.226404 | 24.204086 |
| 60 | 26.091063 | 13.834405 | 27.598576 | 11.834374 | 16.222889 | 12.072435 | 18.653923 | 29.075808 | 21.415884 |
| 70 | 23.424873 | 11.222336 | 25.771370 | 9,565917 | 12.826325 | 9,732855 | 13.965673 | 27.762875 | 18.266953 |
| 80 | 18.892735 | 9,078694 | 23.669950 | 7,751638 | 9,677201 | 7,851287 | 10.155648 | 26.217470 | 14.728153 |
| 90 | 12.111498 | 7,436820 | 20.484146 | 6,325416 | 6,913363 | 6,374051 | 7,291988 | 23.962019 | 10.938133 |

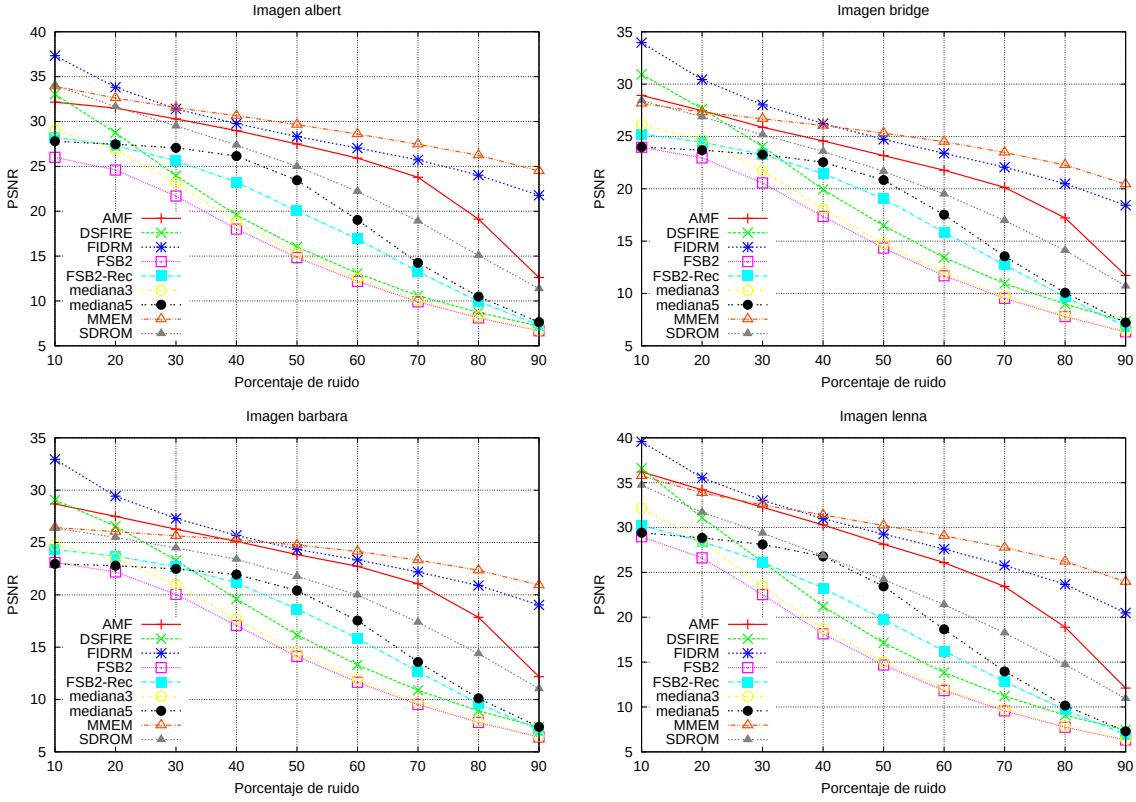



Figura 5.32: Comparación gráfica del parámetro PSNR para los distintos métodos de la literatura.

además no está específicamente diseñado para el ruido "Salt&Pepper". El método AMF (Adaptative Median Filter) también da buenos resultados aunque siempre por debajo de los anteriores. Por último destacar el filtro SDROM que fue, en su momento, un gran avance y que mantiene también buenos resultados pero ya alejado de los anteriores, sobre todo para ruidos altos.

4.- No todos los filtros basados en lógica difusa dan buenos resultados como puede verse en los casos de filtros DSFIRE o FSB-2 que incluso pueden estar por debajo de los de mediana clásicos.

Basándonos en estas consideraciones, y para posteriores comparaciones, se tomarán como modelo los filtros MMEM, FIDRM, SDROM y AMF.

Para ilustrar los resultados numéricos presentados anteriormente, en las figuras 5.33 y 5.34 se presentan los resultados de aplicación de los métodos elegidos en la literatura, sobre la imagen Albert con un 30 % y un 90 % de ruido añadido. En el caso del ruido 30 % no se aprecian diferencias importantes y la reconstrucción es más o menos buena. Sin embargo, con un 90 % los resultados son muy diferentes y resulta sorprendente que con el método MMEM se pueda reconstruir una imagen que ni siquiera se adivina con la gran cantidad de ruido añadido.

**Métodos de mediana.** En el apartado 5.2.3 se definieron los métodos SVM-Mediana-1 y SVM-Mediana-2 así como una alternativa basada en las RBF. Estos métodos se pueden aplicar tanto de manera recursiva como no recursiva. En el entrenamiento se han fijado tamaños de 32 y ruido añadido del 40 %. Esto hace que tengamos que comparar 6 métodos distintos de las misma forma que en el punto anterior.

Del estudio de las tablas 5.3 y 5.4 y de las figuras 5.35 y 5.36 se puede llegar a las siguientes conclusiones:

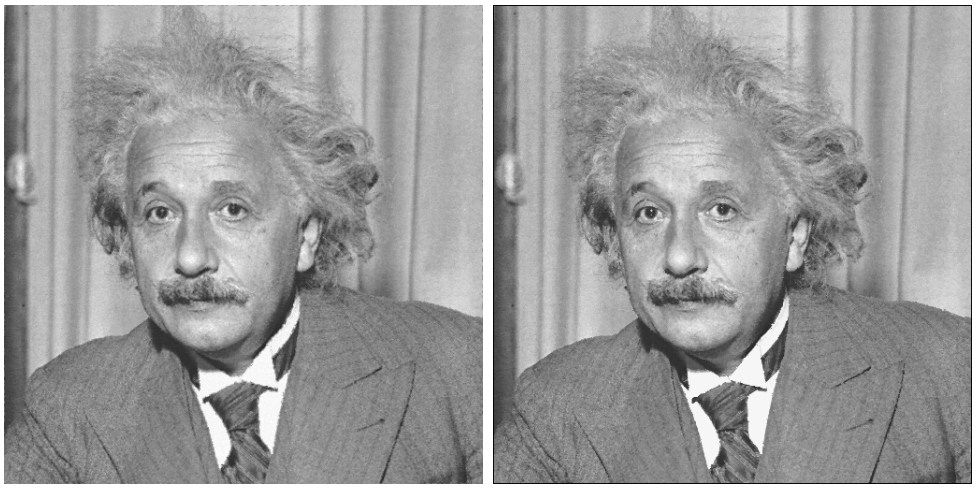
1.- El método SVM-Mediana-2 y su equivalente con RBF se muestran superiores al SVM-Mediana-1 tanto recursivo como no recursivo.

2.- El uso de RBF se muestra inferior en casi todos los casos, pero sorprendentemente mejora a las SVM para el Método-2 (aunque la diferencia es mínima).

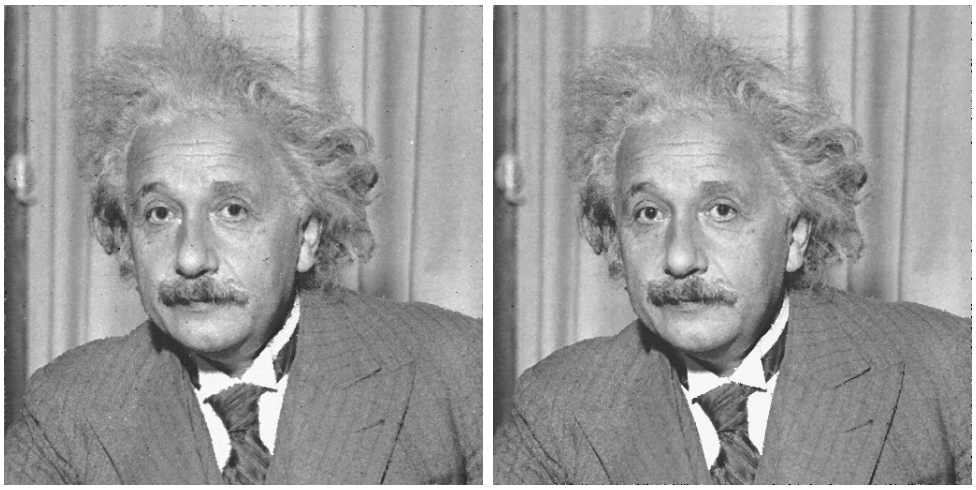
3.- El uso de la recursividad es mejor para tasas de ruido bajas (hasta 50 %). Cuando se superan esas tasas es mejor una solución no recursiva a costa de un mayor coste computacional.

Con esas conclusiones podemos decir que si queremos alta calidad para casi todas las tasas de ruido la mejor opción es el Método-2, para ruidos bajos mejor usar métodos recursivos y para altos no recursivos. Esto no supone desventaja entre los métodos sino que con las herramientas de clasificación que se han diseñado en este capítulo se puede saber tras una primera inspección el nivel aproximado de ruido de la imagen; esa información puede ser usada para elegir el método de filtrado más adecuado, recursivo hasta 50 % o no recursivo para el resto.

Las RBF se muestran como una alternativa adecuada pero hay que recordar que el entrenamiento es más complicado y menos fiable.



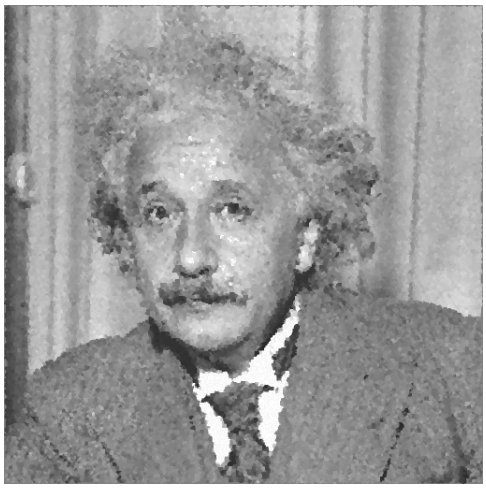
(a) ***MMEM*** (b) ***FIDRM***



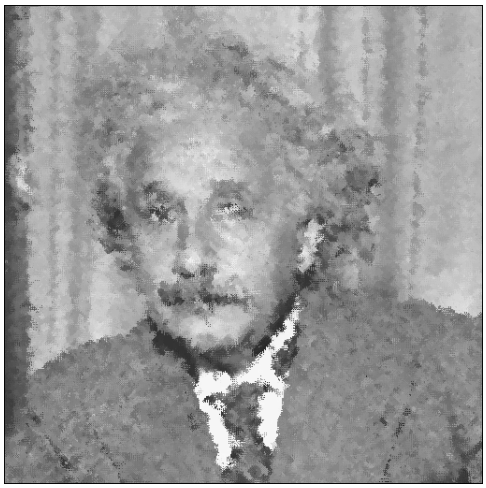
(c) ***SDROM*** (d) ***AMF***

(e) ***Imagen albert con 30 % de ruido***

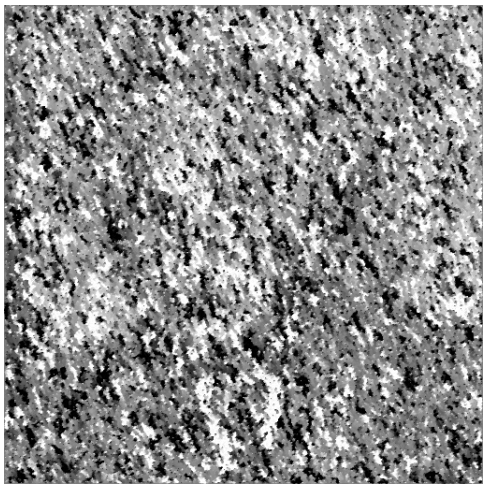
Figura 5.33: Ejemplos de aplicación de algunos métodos de la literatura sobre un ruido moderado.



(a) ***MMEM***

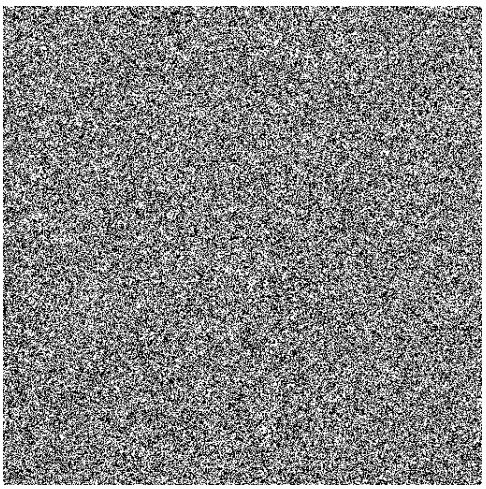


(b) ***FIDRM***



(c) ***SDROM***

(d) ***AMF***



(e) ***Imagen albert con 90 % de ruido***

Figura 5.34: Ejemplos de aplicación de algunos métodos de la literatura sobre un ruido muy alto.

Tabla 5.3: Tablas de resultado MSSIM para los distintos métodos de mediana.

| Imagen albert | | | | | | |
|---|---|---|---|---|---|---|
| | RBF-mediana1-032-040 | RBF-mediana1-nr-032-040 | RBF-mediana2-032-040 | SVM-mediana1-032-040 | SVM-mediana1-nr-032-040 | SVM-mediana2-032-040 |
| 10 | 0.968573 | 0.954228 | 0.968333 | 0.970355 | 0.969856 | 0.969477 |
| 20 | 0.934389 | 0.912375 | 0.934524 | 0.937783 | 0.932756 | 0.934128 |
| 30 | 0.892545 | 0.859907 | 0.895506 | 0.897848 | 0.883512 | 0.890888 |
| 40 | 0.842227 | 0.798288 | 0.851746 | 0.850421 | 0.826111 | 0.841095 |
| 50 | 0.771564 | 0.727403 | 0.798709 | 0.784239 | 0.757366 | 0.783300 |
| 60 | 0.672528 | 0.646627 | 0.738516 | 0.693372 | 0.678491 | 0.720403 |
| 70 | 0.517051 | 0.554096 | 0.668022 | 0.549554 | 0.584245 | 0.653245 |
| 80 | 0.143394 | 0.361067 | 0.590845 | 0.114647 | 0.455690 | 0.574872 |
| 90 | 0.082843 | 0.009652 | 0.494820 | 0.095562 | 0.099578 | 0.482877 |

| Imagen bridge | | | | | | |
|---|---|---|---|---|---|---|
| | RBF-mediana1-032-040 | RBF-mediana1-nr-032-040 | RBF-mediana2-032-040 | SVM-mediana1-032-040 | SVM-mediana1-nr-032-040 | SVM-mediana2-032-040 |
| 10 | 0.971171 | 0.940302 | 0.972441 | 0.975597 | 0.974312 | 0.974964 |
| 20 | 0.937232 | 0.891949 | 0.942809 | 0.947108 | 0.941836 | 0.944301 |
| 30 | 0.894588 | 0.831041 | 0.906630 | 0.910039 | 0.895229 | 0.905032 |
| 40 | 0.837126 | 0.761606 | 0.865881 | 0.858627 | 0.836939 | 0.857983 |
| 50 | 0.752513 | 0.682768 | 0.815742 | 0.782605 | 0.764718 | 0.801981 |
| 60 | 0.630988 | 0.584140 | 0.757083 | 0.669821 | 0.672942 | 0.742328 |
| 70 | 0.439143 | 0.458093 | 0.688188 | 0.479843 | 0.547812 | 0.672328 |
| 80 | 0.111630 | 0.283801 | 0.601821 | 0.106541 | 0.379747 | 0.581805 |
| 90 | 0.055047 | 0.010145 | 0.464938 | 0.065342 | 0.070883 | 0.448361 |

| Imagen barbara | | | | | | |
|---|---|---|---|---|---|---|
| | RBF-mediana1-032-040 | RBF-mediana1-nr-032-040 | RBF-mediana2-032-040 | SVM-mediana1-032-040 | SVM-mediana1-nr-032-040 | SVM-mediana2-032-040 |
| 10 | 0.976966 | 0.971859 | 0.977483 | 0.979477 | 0.979195 | 0.978387 |
| 20 | 0.949243 | 0.938749 | 0.952552 | 0.954404 | 0.950723 | 0.951582 |
| 30 | 0.910545 | 0.893881 | 0.922811 | 0.923487 | 0.913084 | 0.918918 |
| 40 | 0.862754 | 0.840957 | 0.889174 | 0.882102 | 0.866648 | 0.879081 |
| 50 | 0.796288 | 0.778694 | 0.848580 | 0.822537 | 0.809744 | 0.834148 |
| 60 | 0.699442 | 0.706475 | 0.802644 | 0.735816 | 0.740928 | 0.785479 |
| 70 | 0.526022 | 0.594084 | 0.747027 | 0.565546 | 0.640319 | 0.728889 |
| 80 | 0.131690 | 0.391288 | 0.680458 | 0.101119 | 0.492124 | 0.656379 |
| 90 | 0.060060 | 0.012511 | 0.576771 | 0.065202 | 0.071196 | 0.555631 |

| Imagen lenna | | | | | | |
|---|---|---|---|---|---|---|
| | RBF-mediana1-032-040 | RBF-mediana1-nr-032-040 | RBF-mediana2-032-040 | SVM-mediana1-032-040 | SVM-mediana1-nr-032-040 | SVM-mediana2-032-040 |
| 10 | 0.983404 | 0.974115 | 0.983838 | 0.985731 | 0.985389 | 0.985597 |
| 20 | 0.963215 | 0.934212 | 0.964634 | 0.968976 | 0.965105 | 0.968768 |
| 30 | 0.935471 | 0.885455 | 0.940913 | 0.947031 | 0.936497 | 0.949175 |
| 40 | 0.894576 | 0.823877 | 0.911972 | 0.915821 | 0.898262 | 0.926077 |
| 50 | 0.834713 | 0.754516 | 0.876038 | 0.865278 | 0.849831 | 0.896757 |
| 60 | 0.726943 | 0.667571 | 0.834369 | 0.782556 | 0.782010 | 0.860513 |
| 70 | 0.558062 | 0.574193 | 0.784754 | 0.623043 | 0.689169 | 0.820480 |
| 80 | 0.205084 | 0.419624 | 0.708367 | 0.073304 | 0.550212 | 0.767680 |
| 90 | 0.068875 | 0.009347 | 0.599940 | 0.047179 | 0.050093 | 0.687239 |

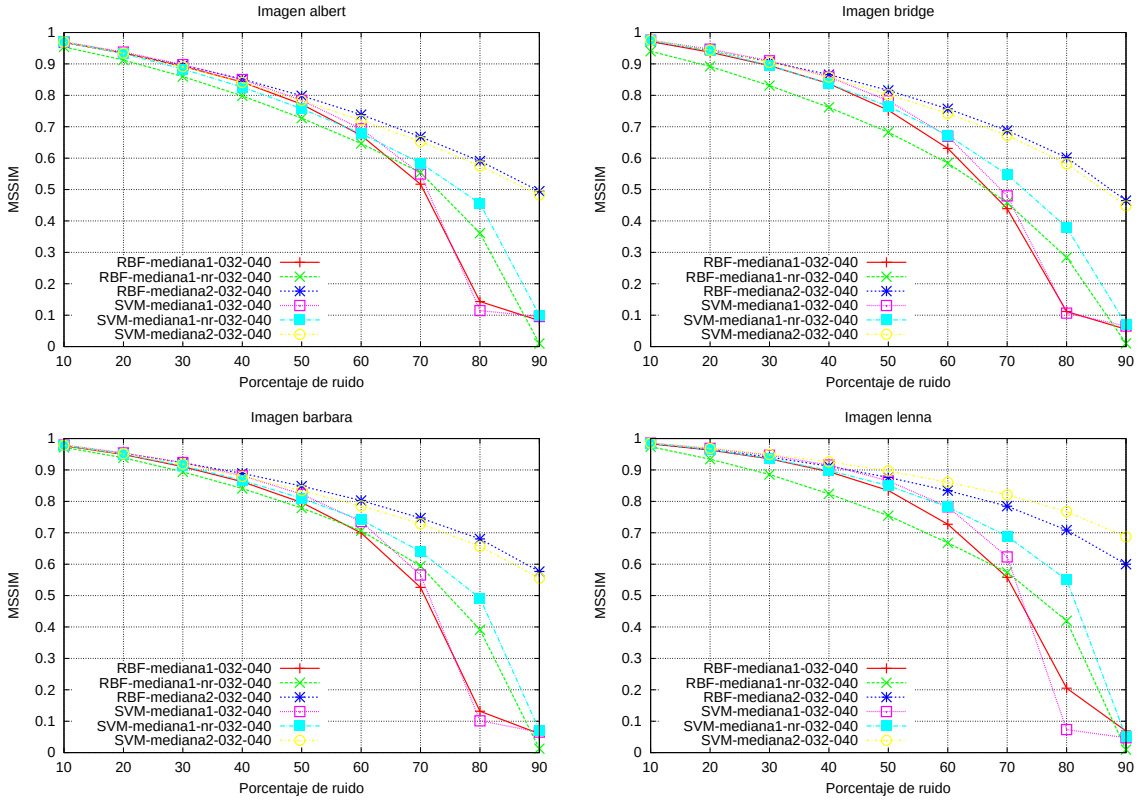



Figura 5.35: Comparación gráfica del parámetro MSSIM para los métodos de mediana.

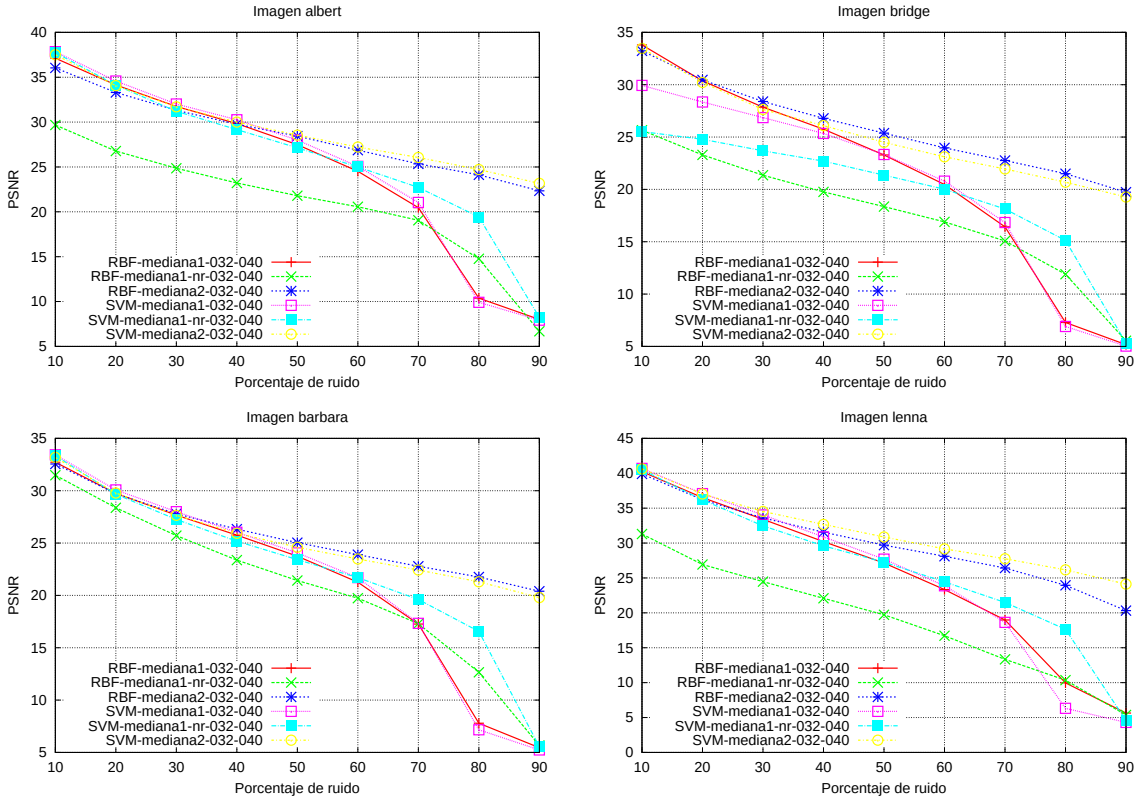



Figura 5.36: Comparación gráfica del parámetro PSNR para los métodos de mediana.

Tabla 5.4: Tablas de resultado PSNR para los distintos métodos de mediana.

| Imagen albert | | | | | | |
|---|---|---|---|---|---|---|
| | RBF-mediana1-032-040 | RBF-mediana1-nr-032-040 | RBF-mediana2-032-040 | SVM-mediana1-032-040 | SVM-mediana1-nr-032-040 | SVM-mediana2-032-040 |
| 10 | 37.118702 | 29.675655 | 36.038845 | 37.849239 | 37.741020 | 37.500107 |
| 20 | 34.143841 | 26.769209 | 33.300739 | 34.576908 | 34.123886 | 34.058155 |
| 30 | 31.736818 | 24.857779 | 31.313227 | 32.018955 | 31.207706 | 31.708488 |
| 40 | 29.829214 | 23.209238 | 29.760592 | 30.264290 | 29.160055 | 30.002531 |
| 50 | 27.488766 | 21.797499 | 28.443613 | 27.992619 | 27.128786 | 28.550467 |
| 60 | 24.512005 | 20.562056 | 26.874748 | 25.065483 | 24.999554 | 27.214975 |
| 70 | 20.468458 | 19.061428 | 25.347881 | 21.047077 | 22.727390 | 26.059565 |
| 80 | 10.378245 | 14.767505 | 24.119425 | 9.910372 | 19.425909 | 24.710365 |
| 90 | 8.009372 | 6.687783 | 22.337889 | 7.922991 | 8.233385 | 23.165743 |

| Imagen bridge | | | | | | |
|---|---|---|---|---|---|---|
| | RBF-mediana1-032-040 | RBF-mediana1-nr-032-040 | RBF-mediana2-032-040 | SVM-mediana1-032-040 | SVM-mediana1-nr-032-040 | SVM-mediana2-032-040 |
| 10 | 33.797039 | 25.646528 | 33.235035 | 29.929312 | 25.520927 | 33.467106 |
| 20 | 30.368292 | 23.294731 | 30.459459 | 28.348816 | 24.782522 | 30.216669 |
| 30 | 27.850664 | 21.352947 | 28.393957 | 26.858362 | 23.686195 | 27.627403 |
| 40 | 25.747458 | 19.761852 | 26.785297 | 25.345196 | 22.701221 | 26.095036 |
| 50 | 23.290375 | 18.356716 | 25.373844 | 23.331079 | 21.360931 | 24.494541 |
| 60 | 20.464020 | 16.897457 | 23.966379 | 20.785084 | 20.017977 | 23.112602 |
| 70 | 16.423651 | 15.074669 | 22.759432 | 16.833603 | 18.124243 | 21.950577 |
| 80 | 7.289472 | 11.895858 | 21.501402 | 6.906212 | 15.120926 | 20.690088 |
| 90 | 5.166737 | 5.546515 | 19.728746 | 5.034148 | 5.320448 | 19.276445 |

| Imagen barbara | | | | | | |
|---|---|---|---|---|---|---|
| | RBF-mediana1-032-040 | RBF-mediana1-nr-032-040 | RBF-mediana2-032-040 | SVM-mediana1-032-040 | SVM-mediana1-nr-032-040 | SVM-mediana2-032-040 |
| 10 | 32.724819 | 31.477962 | 32.550171 | 33.432308 | 33.345146 | 33.180340 |
| 20 | 29.751291 | 28.370399 | 29.715727 | 30.079029 | 29.662218 | 29.778234 |
| 30 | 27.668274 | 25.707874 | 27.795553 | 27.995998 | 27.258312 | 27.650928 |
| 40 | 25.755693 | 23.371483 | 26.331572 | 26.033045 | 25.172085 | 25.933958 |
| 50 | 23.747709 | 21.422817 | 25.039684 | 24.056742 | 23.419893 | 24.603617 |
| 60 | 21.260847 | 19.733030 | 23.890154 | 21.693623 | 21.696049 | 23.480724 |
| 70 | 17.264526 | 17.298946 | 22.783840 | 17.334358 | 19.611546 | 22.424166 |
| 80 | 7.787752 | 12.654865 | 21.755035 | 7.161861 | 16.560051 | 21.296112 |
| 90 | 5.454126 | 5.668634 | 20.398773 | 5.234763 | 5.550314 | 19.791422 |

| Imagen lenna | | | | | | |
|---|---|---|---|---|---|---|
| | RBF-mediana1-032-040 | RBF-mediana1-nr-032-040 | RBF-mediana2-032-040 | SVM-mediana1-032-040 | SVM-mediana1-nr-032-040 | SVM-mediana2-032-040 |
| 10 | 40.223640 | 31.297970 | 39.904686 | 40.698692 | 40.503536 | 40.625160 |
| 20 | 36.514362 | 26.927706 | 36.200989 | 37.084255 | 36.181671 | 37.025070 |
| 30 | 33.403923 | 24.472727 | 33.595310 | 34.020573 | 32.466667 | 34.533054 |
| 40 | 30.204956 | 22.082167 | 31.561974 | 30.877758 | 29.594528 | 32.670368 |
| 50 | 27.149137 | 19.728889 | 29.705770 | 27.699474 | 27.260374 | 30.876043 |
| 60 | 23.336752 | 16.715250 | 28.105936 | 23.870687 | 24.441532 | 29.167273 |
| 70 | 18.982201 | 13.335686 | 26.425007 | 18.630854 | 21.491413 | 27.747007 |
| 80 | 9.963724 | 10.352542 | 23.947630 | 6.319733 | 17.639154 | 26.155390 |
| 90 | 5.642506 | 5.370746 | 20.321136 | 4.301896 | 4.577703 | 24.111965 |

(a) ***RBF Mediana-1***

(b) ***RBF Mediana-1 No recursiva***

(c) ***RBF Mediana-2***

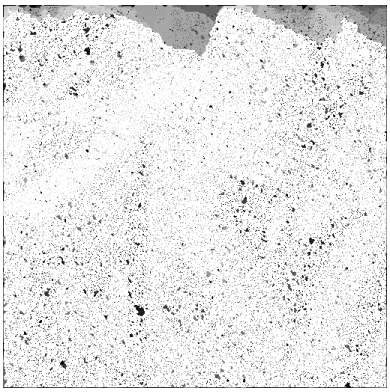
(d) ***SVM Mediana-1***

(e) ***SVM Mediana-1 No recursiva***

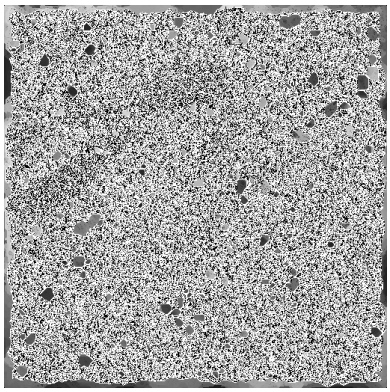
(f) ***SVM Mediana-2***

(g) ***Imagen barbara con 30 % de ruido***

Figura 5.37: Ejemplos de aplicación de algunos métodos de mediana modificados sobre un ruido moderado.

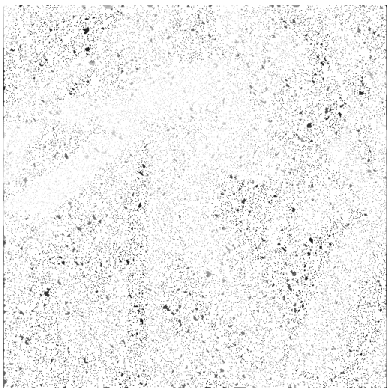


(a) ***RBF Mediana-1***

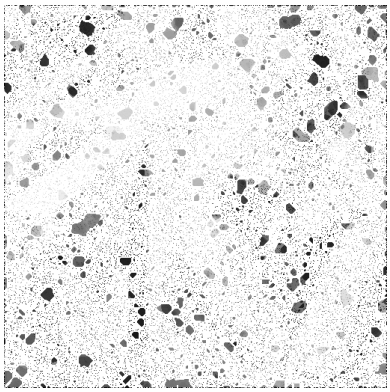


(b) ***RBF Mediana-1 No recursiva***

(c) ***RBF Mediana-2***

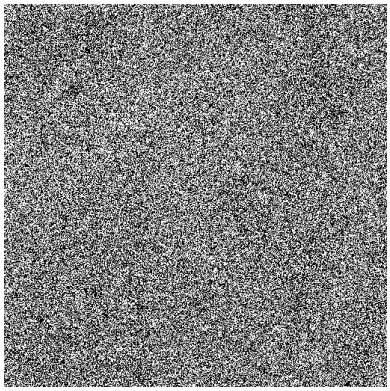


(d) ***SVM Mediana-1***



(e) ***SVM Mediana-1 No recursiva***

(f) ***SVM Mediana-2***



(g) ***Imagen barbara con 90 % de ruido***

Figura 5.38: Ejemplos de aplicación de algunos métodos de mediana modificados sobre un ruido muy alto.

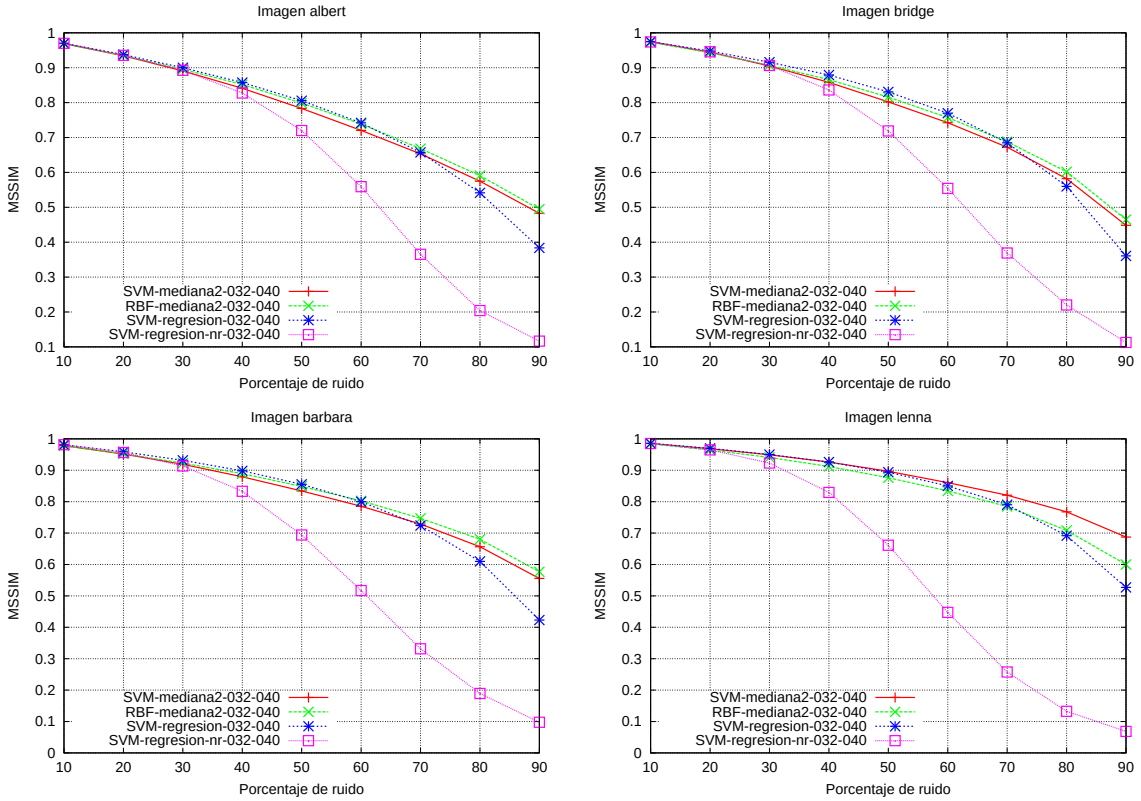



Figura 5.39: Comparación gráfica del parámetro MSSIM para los métodos de mediana y regresión.

En las figuras 5.37 y 5.38 se muestran algunos resultados de la aplicación de los filtros de mediana modificados. En el caso de la figura 5.37 se aprecia nuevamente que la reconstrucción es buena pero es de destacar en la imagen de la figura 5.37(b) que algunos píxeles ruidosos en negro aparecen debido a una mala clasificación. Con el ruido muy alto de la figura 5.38 solo los métodos con Mediana-2 son capaces de reconstruir la imagen y además lo hacen de una manera muy efectiva.

**Comparación del método de regresión.** En este apartado se comparará el método de eliminación de ruido que no utiliza la mediana sino la regresión con SVM con los mejores de los que sí usan la mediana, concretamente los SVM-Mediana-2 y RBF-Mediana-2. Siendo el método de regresión una implementación recursiva en su idea inicial pareció conveniente también el intento de aplicarlo de manera no recursiva para observar si existía alguna mejora. En las tablas 5.5, 5.6 y en las figuras 5.39 y 5.40 están reflejados los datos obtenidos para estos métodos.

El análisis de los datos nos ofrece algunas conclusiones:

1.- El método de regresión no recursivo es muy inferior a los otros comparados. Sólo para porcentajes de ruido pequeños los resultados son comparables a los otros métodos pero su funcionamiento es muy inferior según el ruido aumenta.

2.- El método de regresión es superior a los de mediana para porcentajes de ruido hasta 60 ó 70 %. Si la comparación se realiza en MSSIM la diferencia es muy pequeña y un

Tabla 5.5: Tablas de resultado MSSIM para los distintos métodos de mediana y regresión.

| Imagen albert | | | | |
|---|---|---|---|---|
| | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 | SVM-regresion-nr-032-040 |
| 10 | 0,969477 | 0,968333 | 0,970241 | 0,969928 |
| 20 | 0,934128 | 0,934524 | 0,937615 | 0,935639 |
| 30 | 0,890888 | 0,895506 | 0,899937 | 0,892671 |
| 40 | 0,841095 | 0,851746 | 0,857560 | 0,827534 |
| 50 | 0,783300 | 0,798709 | 0,805460 | 0,719884 |
| 60 | 0,720403 | 0,738516 | 0,741865 | 0,559200 |
| 70 | 0,653245 | 0,668022 | 0,657270 | 0,365407 |
| 80 | 0,574872 | 0,590845 | 0,541535 | 0,204052 |
| 90 | 0,482877 | 0,494820 | 0,383766 | 0,116416 |

| Imagen bridge | | | | |
|---|---|---|---|---|
| | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 | SVM-regresion-nr-032-040 |
| 10 | 0,974964 | 0,972441 | 0,974756 | 0,973944 |
| 20 | 0,944301 | 0,942809 | 0,947699 | 0,945635 |
| 30 | 0,905032 | 0,906630 | 0,915737 | 0,906492 |
| 40 | 0,857983 | 0,865881 | 0,879040 | 0,836336 |
| 50 | 0,801981 | 0,815742 | 0,830698 | 0,718785 |
| 60 | 0,742328 | 0,757083 | 0,769699 | 0,554175 |
| 70 | 0,672328 | 0,688188 | 0,684800 | 0,369080 |
| 80 | 0,581805 | 0,601821 | 0,559848 | 0,220134 |
| 90 | 0,448361 | 0,464938 | 0,360792 | 0,112666 |

| Imagen barbara | | | | |
|---|---|---|---|---|
| | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 | SVM-regresion-nr-032-040 |
| 10 | 0,978387 | 0,977483 | 0,981415 | 0,981375 |
| 20 | 0,951582 | 0,952552 | 0,958594 | 0,956424 |
| 30 | 0,918918 | 0,922811 | 0,931576 | 0,913662 |
| 40 | 0,879081 | 0,889174 | 0,898445 | 0,833097 |
| 50 | 0,834148 | 0,848580 | 0,855373 | 0,693824 |
| 60 | 0,785479 | 0,802644 | 0,799991 | 0,517108 |
| 70 | 0,728889 | 0,747027 | 0,723928 | 0,331465 |
| 80 | 0,656379 | 0,680458 | 0,609732 | 0,189358 |
| 90 | 0,555631 | 0,576771 | 0,423135 | 0,097580 |

| Imagen lenna | | | | |
|---|---|---|---|---|
| | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 | SVM-regresion-nr-032-040 |
| 10 | 0,985597 | 0,983838 | 0,985816 | 0,985248 |
| 20 | 0,968768 | 0,964634 | 0,969672 | 0,964897 |
| 30 | 0,949175 | 0,940913 | 0,950045 | 0,922914 |
| 40 | 0,926077 | 0,911972 | 0,926045 | 0,829544 |
| 50 | 0,896757 | 0,876038 | 0,893720 | 0,661212 |
| 60 | 0,860513 | 0,834369 | 0,850272 | 0,447610 |
| 70 | 0,820480 | 0,784754 | 0,790297 | 0,257707 |
| 80 | 0,767680 | 0,708367 | 0,691536 | 0,132520 |
| 90 | 0,687239 | 0,599940 | 0,527008 | 0,068566 |

Tabla 5.6: Tablas de resultado PSNR para los distintos métodos de mediana y regresión.

| Imagen albert | | | | |
|---|---|---|---|---|
| | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 | SVM-regresion-nr-032-040 |
| 10 | 37.500107 | 36.038845 | 36.949211 | 36.836834 |
| 20 | 34.058155 | 33.300739 | 33.687267 | 33.602131 |
| 30 | 31.708488 | 31.313227 | 31.709190 | 31.629862 |
| 40 | 30.002531 | 29.760592 | 29.917974 | 29.426447 |
| 50 | 28.550467 | 28.443613 | 28.458282 | 26.659208 |
| 60 | 27.214975 | 26.874748 | 26.827600 | 23.146530 |
| 70 | 26.059565 | 25.347881 | 25.025618 | 19.322252 |
| 80 | 24.710365 | 24.119425 | 22.280787 | 15.466531 |
| 90 | 23.165743 | 22.337889 | 18.570232 | 11.712202 |

| Imagen bridge | | | | |
|---|---|---|---|---|
| | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 | SVM-regresion-nr-032-040 |
| 10 | 33.467106 | 33.235035 | 32.700069 | 27.727610 |
| 20 | 30.216669 | 30.459459 | 30.206335 | 26.628353 |
| 30 | 27.627403 | 28.393957 | 28.464655 | 25.631899 |
| 40 | 26.095036 | 26.785297 | 27.149557 | 24.065901 |
| 50 | 24.494541 | 25.373844 | 25.770967 | 21.884731 |
| 60 | 23.112602 | 23.966379 | 24.382690 | 19.087490 |
| 70 | 21.950577 | 22.759432 | 22.934708 | 15.862901 |
| 80 | 20.690088 | 21.501402 | 20.945957 | 12.427894 |
| 90 | 19.276445 | 19.728746 | 17.473093 | 8,852635 |

| Imagen barbara | | | | |
|---|---|---|---|---|
| | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 | SVM-regresion-nr-032-040 |
| 10 | 33.180340 | 32.550171 | 34.283176 | 34.327801 |
| 20 | 29.778234 | 29.715727 | 30.941805 | 30.946358 |
| 30 | 27.650928 | 27.795553 | 29.019274 | 28.698706 |
| 40 | 25.933958 | 26.331572 | 27.529587 | 26.230293 |
| 50 | 24.603617 | 25.039684 | 26.152454 | 23.175800 |
| 60 | 23.480724 | 23.890154 | 24.889143 | 19.887997 |
| 70 | 22.424166 | 22.783840 | 23.452574 | 16.310776 |
| 80 | 21.296112 | 21.755035 | 21.495098 | 12.804863 |
| 90 | 19.791422 | 20.398773 | 17.803490 | 9,175740 |

| Imagen lenna | | | | |
|---|---|---|---|---|
| | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 | SVM-regresion-nr-032-040 |
| 10 | 40.625160 | 39.904686 | 40.548576 | 40.388084 |
| 20 | 37.025070 | 36.200989 | 37.097847 | 36.619152 |
| 30 | 34.533054 | 33.595310 | 34.827099 | 32.875275 |
| 40 | 32.670368 | 31.561974 | 32.815884 | 28.243816 |
| 50 | 30.876043 | 29.705770 | 31.061268 | 23.605562 |
| 60 | 29.167273 | 28.105936 | 29.173763 | 19.436386 |
| 70 | 27.747007 | 26.425007 | 26.987381 | 15.672877 |
| 80 | 26.155390 | 23.947630 | 24.472937 | 12.070536 |
| 90 | 24.111965 | 20.321136 | 20.372669 | 8,328642 |

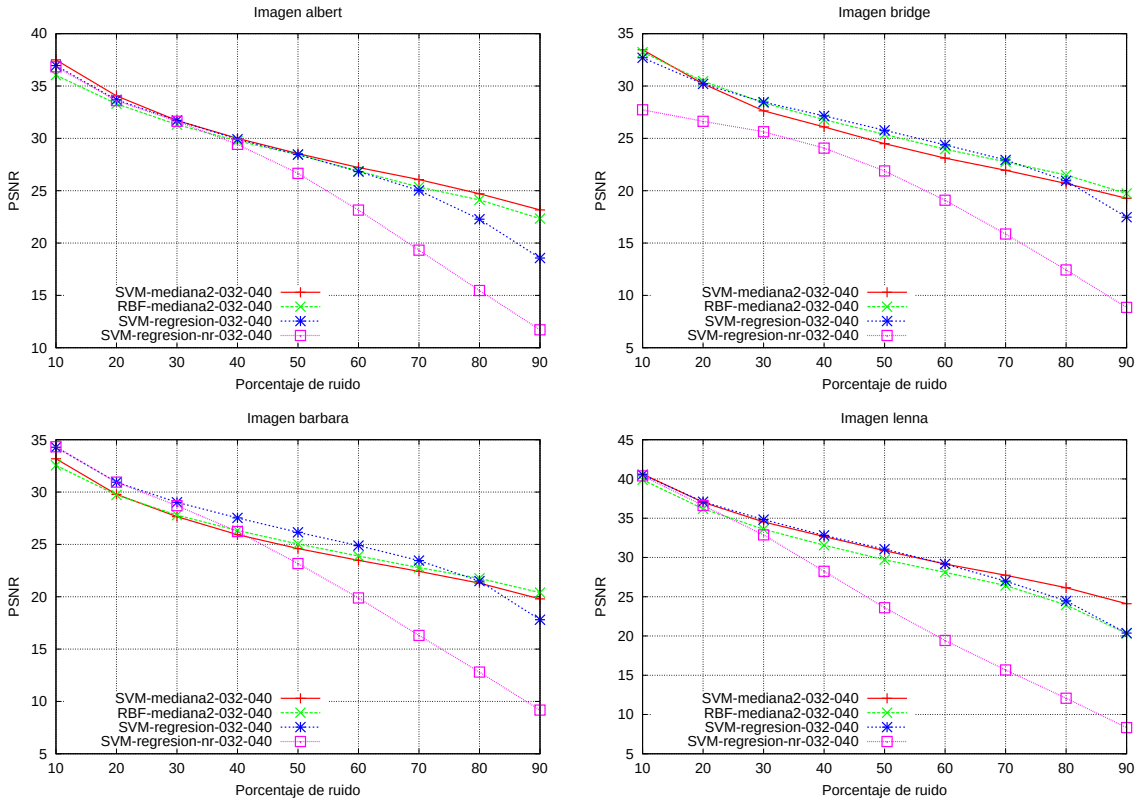



Figura 5.40: Comparación gráfica del parámetro PSNR para los métodos de mediana y regresión.

poco mayor para PSNR. Esta diferencia se agranda para imágenes como Barbara que están más texturadas.

3.- La ganancia conseguida no es excesivamente grande y sólo sería conveniente usar la regresión para el caso de imágenes texturadas.

Para la comparación final se considerará el método de regresión en su versión recursiva puesto que la no recursiva es muy inferior.

En las figuras 5.41 y 5.42 puede apreciarse el efecto de cada uno de los métodos comparados sobre la imagen bridge. Con poco ruido (Figura 5.41) el resultado es nuevamente aceptable para todos los métodos. Los peores efectos se producen con la regresión no recursiva porque incluso quedan píxeles sin reconstruir. Usando la regresión no recursiva aparece un efecto de arrastrado, sobre todo en la zona negra inferior derecha. Este efecto es propio de la regresión y para esta imagen en concreto no representa un problema grave. Con ruido muy alto (Figura 5.42) los defectos señalados anteriormente se acentúan haciendo que la regresión no sea práctica en este caso.

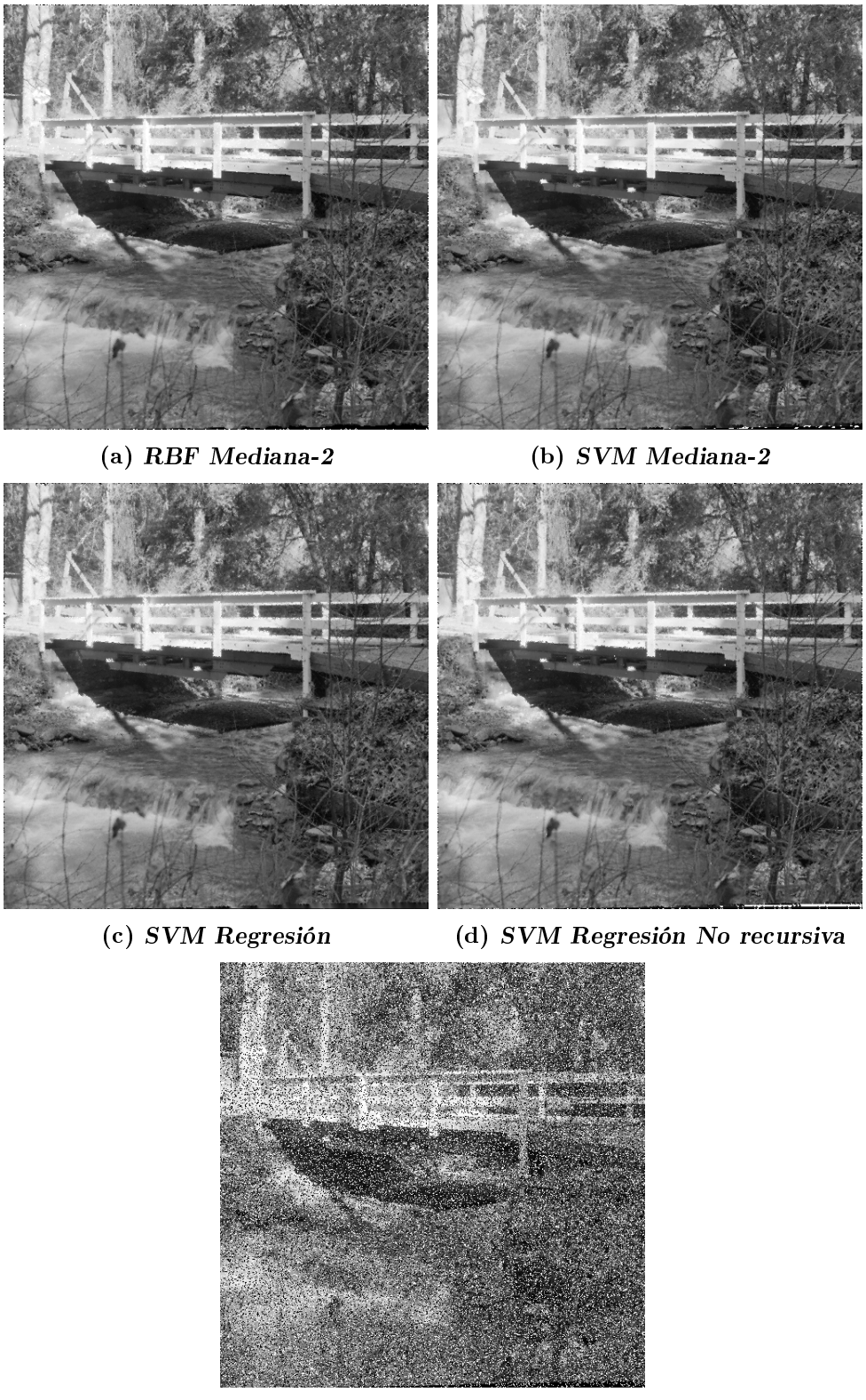
**Comparación de métodos propuestos y en la literatura.** Una vez presentados los resultados de los métodos de la literatura y de los propuestos se compararán para comprobar las mejoras aportadas. La comparación se realizará sólo con aquellos que se han ido seleccionando en los apartados previos.

Estudiando las tablas 5.7 y 5.8 así como las gráficas 5.43 y 5.44 se puede llegar a una serie de conclusiones. Una primera conclusión es que el mejor método cuando el ruido es alto o muy alto es el MSSIM, aunque los resultados dependen bastante del tipo de imagen. Por ejemplo, en barbara es mejorado por la regresión hasta un 70 % de ruido. Por otro lado los resultados de los métodos propuestos en esta tesis son muchas veces comparables, e incluso superiores, a los de los métodos de la literatura y sólo son superados por el método MMEM antes citado.

Además de los datos presentados en las tablas y gráficas se muestran también resultados reales sobre la imagen Lenna en las figuras 5.45 y 5.46. Una vez más las diferencias son mínimas con poco ruido pero llamativas cuando el ruido es muy alto. Hay que destacar como los métodos SVM-Mediana-2 y MMEM consiguen recuperar el contenido de la imagen aun cuando sólo queda un 10 % de la información de la imagen. El método SVM-Mediana-2 se muestra superior en reconstrucción para ruidos hasta el 60 % para las imágenes Lenna y Barbara mientras que para Albert el método mejor es el SVM-regresión. Los métodos propuestos se muestran competitivos para distintos niveles de ruido, lo que los hace más versátiles que métodos establecidos como el MMEM pero que están diseñados para niveles altos de ruido.

## 5.4. Resumen

En este capítulo se han presentado una serie de métodos basados en SVM que permiten la reducción del ruido impulsivo en imágenes en blanco y negro. Por un lado se ha estudiado la capacidad de detección de los píxeles ruidosos tras un entrenamiento sintético y por otro la posibilidad de reconstrucción usando la regresión son SVM con un entrenamiento igualmente sintético.



(a) ***RBF Mediana-2*** (b) ***SVM Mediana-2***

(c) ***SVM Regresión*** (d) ***SVM Regresión No recursiva***

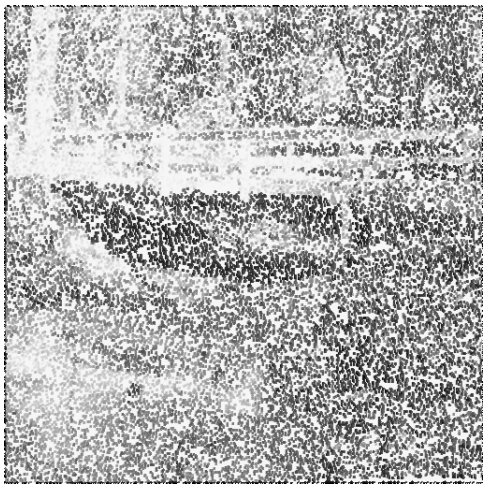
(e) ***Imagen bridge con 30 % de ruido***

Figura 5.41: Ejemplos de aplicación de métodos de mediana modificados y regresión sobre un ruido moderado.

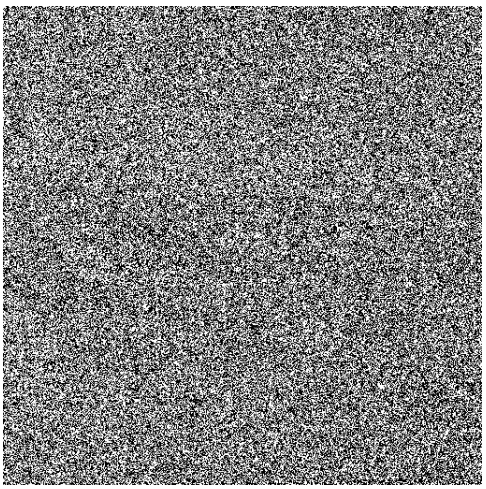
(a) ***RBF Mediana-2***

(b) ***SVM Mediana-2***

(c) ***SVM Regresión***



(d) ***SVM Regresión No recursiva***



(e) ***Imagen bridge con 90 % de ruido***

Figura 5.42: Ejemplos de aplicación de métodos de mediana modificados y regresión sobre un ruido muy alto.

Tabla 5.7: Tablas de resultado MSSIM para comparar métodos propuestos y en la literatura.

| Imagen albert | | | | | | | |
|---|---|---|---|---|---|---|---|
| | MMEM | AMF | FIDRM | SDROM | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 |
| 10 | 0.956672 | 0.868121 | 0.967178 | 0.950471 | 0.969477 | 0.968333 | 0.970241 |
| 20 | 0.926977 | 0.862645 | 0.928897 | 0.907875 | 0.934128 | 0.934524 | 0.937615 |
| 30 | 0.894313 | 0.837262 | 0.881967 | 0.852711 | 0.890888 | 0.895506 | 0.899937 |
| 40 | 0.858486 | 0.799501 | 0.830517 | 0.776943 | 0.841095 | 0.851746 | 0.857560 |
| 50 | 0.815968 | 0.748164 | 0.771919 | 0.669926 | 0.783300 | 0.798709 | 0.805460 |
| 60 | 0.765509 | 0.683942 | 0.708554 | 0.531835 | 0.720403 | 0.738516 | 0.741865 |
| 70 | 0.704465 | 0.596994 | 0.635423 | 0.354679 | 0.653245 | 0.668022 | 0.657270 |
| 80 | 0.632327 | 0.419037 | 0.537767 | 0.182095 | 0.574872 | 0.590845 | 0.541535 |
| 90 | 0.537140 | 0.124649 | 0.416417 | 0.067399 | 0.482877 | 0.494820 | 0.383766 |

| Imagen bridge | | | | | | | |
|---|---|---|---|---|---|---|---|
| | MMEM | AMF | FIDRM | SDROM | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 |
| 10 | 0.925546 | 0.908810 | 0.973761 | 0.914898 | 0.974964 | 0.972441 | 0.974756 |
| 20 | 0.900200 | 0.892997 | 0.939651 | 0.870606 | 0.944301 | 0.942809 | 0.947699 |
| 30 | 0.873745 | 0.861833 | 0.896047 | 0.812396 | 0.905032 | 0.906630 | 0.915737 |
| 40 | 0.846285 | 0.818482 | 0.845570 | 0.739280 | 0.857983 | 0.865881 | 0.879040 |
| 50 | 0.811575 | 0.761476 | 0.785036 | 0.632748 | 0.801981 | 0.815742 | 0.830698 |
| 60 | 0.768116 | 0.691620 | 0.718108 | 0.500707 | 0.742328 | 0.757083 | 0.769699 |
| 70 | 0.711744 | 0.590708 | 0.630632 | 0.345197 | 0.672328 | 0.688188 | 0.684800 |
| 80 | 0.633920 | 0.416639 | 0.513291 | 0.187178 | 0.581805 | 0.601821 | 0.559848 |
| 90 | 0.499710 | 0.134509 | 0.346012 | 0.067004 | 0.448361 | 0.464938 | 0.360792 |

| Imagen barbara | | | | | | | |
|---|---|---|---|---|---|---|---|
| | MMEM | AMF | FIDRM | SDROM | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 |
| 10 | 0.896966 | 0.929117 | 0.975972 | 0.889462 | 0.978387 | 0.977483 | 0.981415 |
| 20 | 0.877041 | 0.911970 | 0.946770 | 0.845796 | 0.951582 | 0.952552 | 0.958594 |
| 30 | 0.855632 | 0.884692 | 0.910910 | 0.789102 | 0.918918 | 0.922811 | 0.931576 |
| 40 | 0.835505 | 0.850235 | 0.869650 | 0.722051 | 0.879081 | 0.889174 | 0.898445 |
| 50 | 0.809643 | 0.804300 | 0.822102 | 0.616627 | 0.834148 | 0.848580 | 0.855373 |
| 60 | 0.779711 | 0.751471 | 0.772958 | 0.489460 | 0.785479 | 0.802644 | 0.799991 |
| 70 | 0.738071 | 0.665855 | 0.703291 | 0.328331 | 0.728889 | 0.747027 | 0.723928 |
| 80 | 0.681819 | 0.481255 | 0.608531 | 0.178648 | 0.656379 | 0.680458 | 0.609732 |
| 90 | 0.588916 | 0.147859 | 0.468699 | 0.070370 | 0.555631 | 0.576771 | 0.423135 |

| Imagen lenna | | | | | | | |
|---|---|---|---|---|---|---|---|
| | MMEM | AMF | FIDRM | SDROM | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 |
| 10 | 0.965510 | 0.943772 | 0.982776 | 0.957541 | 0.985597 | 0.983838 | 0.985816 |
| 20 | 0.940751 | 0.934364 | 0.961310 | 0.917074 | 0.968768 | 0.964634 | 0.969672 |
| 30 | 0.917256 | 0.916601 | 0.935665 | 0.864853 | 0.949175 | 0.940913 | 0.950045 |
| 40 | 0.893362 | 0.889887 | 0.905280 | 0.788234 | 0.926077 | 0.911972 | 0.926045 |
| 50 | 0.867704 | 0.850511 | 0.868523 | 0.673200 | 0.896757 | 0.876038 | 0.893720 |
| 60 | 0.837583 | 0.801044 | 0.825154 | 0.528740 | 0.860513 | 0.834369 | 0.850272 |
| 70 | 0.801534 | 0.721114 | 0.768082 | 0.359055 | 0.820480 | 0.784754 | 0.790297 |
| 80 | 0.750824 | 0.524986 | 0.680806 | 0.182144 | 0.767680 | 0.708367 | 0.691536 |
| 90 | 0.671322 | 0.144542 | 0.544621 | 0.066667 | 0.687239 | 0.599940 | 0.527008 |

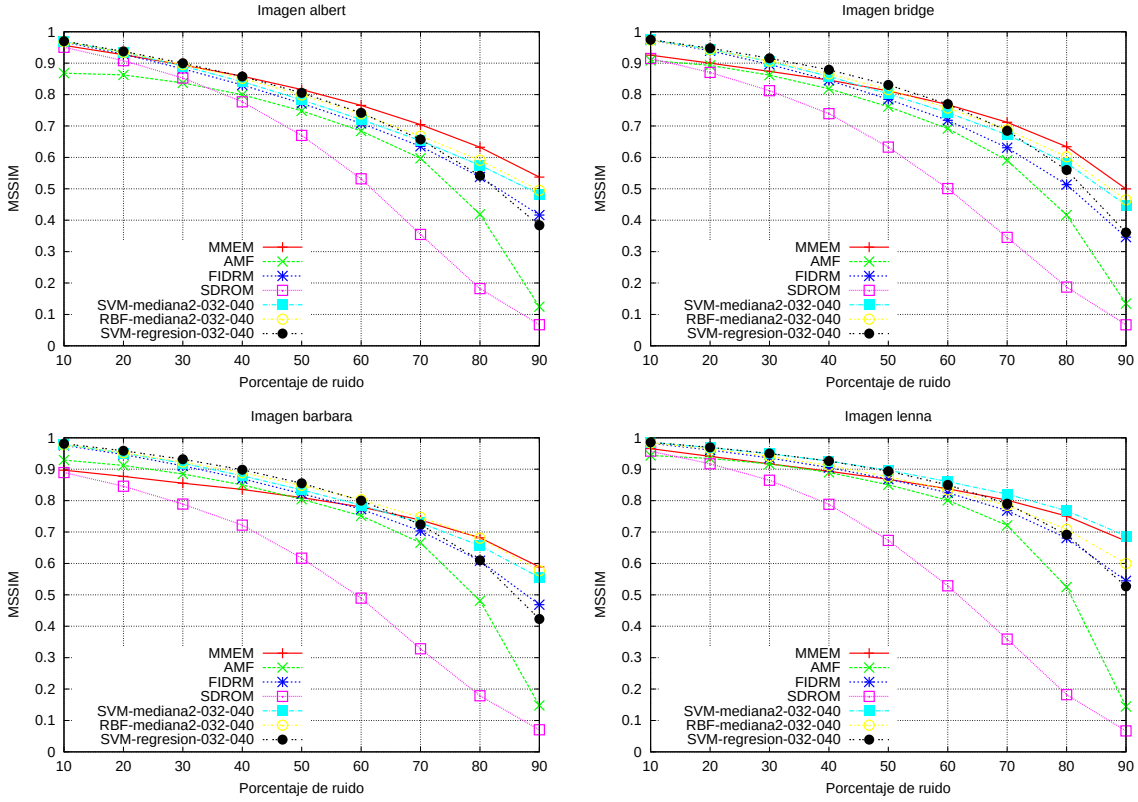



Figura 5.43: Comparación gráfica del parámetro MSSIM de métodos propuestos y en la literatura.

Tabla 5.8: Tablas de resultado PSNR para comparar métodos propuestos y en la literatura.

| Imagen albert | | | | | | | |
|---|---|---|---|---|---|---|---|
| | MMEM | AMF | FIDRM | SDROM | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 |
| 10 | 33.922028 | 32.156223 | 37.333382 | 34.037266 | 37.500107 | 36.038845 | 36.949211 |
| 20 | 32.612499 | 31.494225 | 33.819706 | 31.648333 | 34.058155 | 33.300739 | 33.687267 |
| 30 | 31.541697 | 30.287420 | 31.393667 | 29.499716 | 31.708488 | 31.313227 | 31.709190 |
| 40 | 30.642998 | 28.988165 | 29.766779 | 27.370152 | 30.002531 | 29.760592 | 29.917974 |
| 50 | 29.655785 | 27.496731 | 28.324076 | 24.988792 | 28.550467 | 28.443613 | 28.458282 |
| 60 | 28.604557 | 25.917393 | 27.036194 | 22.217510 | 27.214975 | 26.874748 | 26.827600 |
| 70 | 27.454901 | 23.785048 | 25.716215 | 18.909363 | 26.059565 | 25.347881 | 25.025618 |
| 80 | 26.237661 | 19.116053 | 24.014885 | 15.080471 | 24.710365 | 24.119425 | 22.280787 |
| 90 | 24.485977 | 12.607323 | 21.780247 | 11.367532 | 23.165743 | 22.337889 | 18.570232 |

| Imagen bridge | | | | | | | |
|---|---|---|---|---|---|---|---|
| | MMEM | AMF | FIDRM | SDROM | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 |
| 10 | 28.158722 | 28.918736 | 33.963520 | 28.426336 | 33.467106 | 33.235035 | 32.700069 |
| 20 | 27.326160 | 27.458090 | 30.437441 | 26.883114 | 30.216669 | 30.459459 | 30.206335 |
| 30 | 26.683672 | 25.876610 | 28.022972 | 25.150335 | 27.627403 | 28.393957 | 28.464655 |
| 40 | 26.050400 | 24.572765 | 26.230507 | 23.579012 | 26.095036 | 26.785297 | 27.149557 |
| 50 | 25.321407 | 23.170425 | 24.720730 | 21.657101 | 24.494541 | 25.373844 | 25.770967 |
| 60 | 24.499048 | 21.776451 | 23.397732 | 19.504391 | 23.112602 | 23.966379 | 24.382690 |
| 70 | 23.470270 | 20.127882 | 22.064522 | 16.967669 | 21.950577 | 22.759432 | 22.934708 |
| 80 | 22.280384 | 17.213848 | 20.487640 | 14.115790 | 20.690088 | 21.501402 | 20.945957 |
| 90 | 20.433199 | 11.722597 | 18.422403 | 10.699737 | 19.276445 | 19.728746 | 17.473093 |

| Imagen barbara | | | | | | | |
|---|---|---|---|---|---|---|---|
| | MMEM | AMF | FIDRM | SDROM | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 |
| 10 | 26.428156 | 28.683268 | 32.939953 | 26.378366 | 33.180340 | 32.550171 | 34.283176 |
| 20 | 26.032326 | 27.489840 | 29.426489 | 25.491571 | 29.778234 | 29.715727 | 30.941805 |
| 30 | 25.635643 | 26.269314 | 27.290861 | 24.479036 | 27.650928 | 27.795553 | 29.019274 |
| 40 | 25.260748 | 25.104719 | 25.680399 | 23.404718 | 25.933958 | 26.331572 | 27.529587 |
| 50 | 24.752451 | 23.845617 | 24.345676 | 21.769831 | 24.603617 | 25.039684 | 26.152454 |
| 60 | 24.142450 | 22.729691 | 23.339054 | 19.995382 | 23.480724 | 23.890154 | 24.889143 |
| 70 | 23.317726 | 21.060757 | 22.186180 | 17.388323 | 22.424166 | 22.783840 | 23.452574 |
| 80 | 22.343952 | 17.855021 | 20.894852 | 14.370651 | 21.296112 | 21.755035 | 21.495098 |
| 90 | 20.923389 | 12.178355 | 19.036133 | 11.035494 | 19.791422 | 20.398773 | 17.803490 |

| Imagen lenna | | | | | | | |
|---|---|---|---|---|---|---|---|
| | MMEM | AMF | FIDRM | SDROM | SVM-mediana2-032-040 | RBF-mediana2-032-040 | SVM-regresion-032-040 |
| 10 | 35.750389 | 36.202995 | 39.575054 | 34.718796 | 40.625160 | 39.904686 | 40.548576 |
| 20 | 33.860439 | 34.203274 | 35.537968 | 31.704111 | 37.025070 | 36.200989 | 37.097847 |
| 30 | 32.523937 | 32.254276 | 33.024055 | 29.379898 | 34.533054 | 33.595310 | 34.827099 |
| 40 | 31.411776 | 30.297773 | 30.987045 | 26.858957 | 32.670368 | 31.561974 | 32.815884 |
| 50 | 30.226404 | 28.139463 | 29.266886 | 24.204086 | 30.876043 | 29.705770 | 31.061268 |
| 60 | 29.075808 | 26.091063 | 27.598576 | 21.415884 | 29.167273 | 28.105936 | 29.173763 |
| 70 | 27.762875 | 23.424873 | 25.771370 | 18.266953 | 27.747007 | 26.425007 | 26.987381 |
| 80 | 26.217470 | 18.892735 | 23.669950 | 14.728153 | 26.155390 | 23.947630 | 24.472937 |
| 90 | 23.962019 | 12.111498 | 20.484146 | 10.938133 | 24.111965 | 20.321136 | 20.372669 |

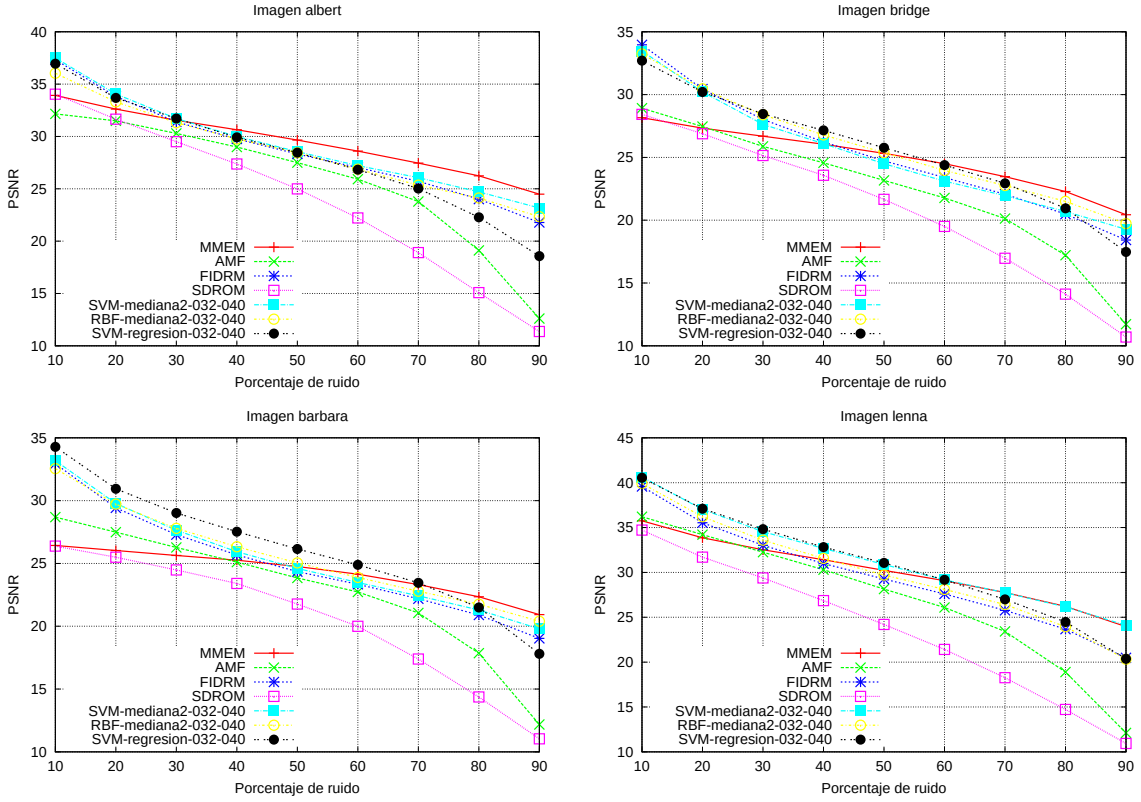



Figura 5.44: Comparación gráfica del parámetro PSNR de métodos propuestos y en la literatura.

(a) ***MMEM*** (b) ***RBF Mediana-2***

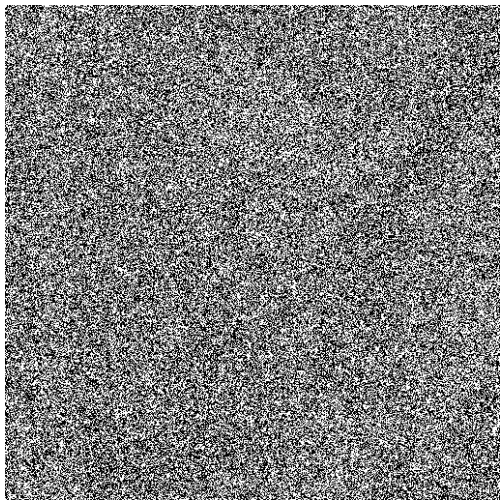
(c) ***SVM Mediana-2*** (d) ***SVM Regresión***

(e) ***Imagen lenna con 30 % de ruido***

Figura 5.45: Ejemplos de aplicación de los mejores métodos presentados sobre un ruido moderado.

**(a)** ***MMEM*** **(b)** ***RBF Mediana-2***

**(c)** ***SVM Mediana-2*** **(d)** ***SVM Regresión***



**(e)** ***Imagen lenna con 90 % de ruido***

Figura 5.46: Ejemplos de aplicación de los mejores métodos presentados sobre un ruido muy alto.

Una vez comprobada la capacidad tanto de detección como de reconstrucción de las SVM, se han propuesto varios métodos en los que de alguna manera se implementaba o la detección o la reconstrucción o ambas. Entre estos métodos se han propuesto variaciones como las basadas en la clasificación con RBF. para comprobar si realmente las SVM suponen una buena alternativa. Por último, los métodos propuestos se han comparado entre sí y son métodos de la literatura para comprobar si suponen algún tipo de mejora.

Como se ha podido comprobar en los resultados aportados, la calidad de reconstrucción depende en gran manera de la imagen sobre la que se realiza la operación. Por tanto, los resultados obtenidos sólo dan una idea de que los métodos propuestos son competitivos con los ya establecidos pero no permiten afirmar que serán mejores en todos los casos. Sí que se ha demostrado que son una alternativa válida y que para un rango grande de niveles de ruido pueden incluso sobrepasar a otros métodos de la literatura.

# Capítulo 6

# Conclusiones, contribuciones originales y futuras líneas

A continuación se enumeran las principales conclusiones y contribuciones originales que han sido aportadas en esta tesis. Posteriormente se comentarán las futuras líneas de investigación que quedan abiertas tras el desarrollo de la tesis.

## 6.1. Conclusiones

El objetivo inicial de esta tesis era demostrar la capacidad de los algoritmos de aprendizaje estadístico para ser aplicados en tareas de procesado de imagen de bajo nivel. Para ello se eligieron tres tareas propias de este procesado como son la detección de bordes, la segmentación de imágenes en color o la eliminación de ruido (en este caso impulsivo). De los algoritmos de aprendizaje se han utilizado las SVM por sus especiales cualidades y facilidad de aplicación y entrenamiento aunque no se han dejado de explorar otras posibilidades como las redes neuronales RBF.

En cada una de las aplicaciones en las que las SVM han sido introducidas se ha demostrado su capacidad para obtener buenos resultados y, sobre todo, la posibilidad de adaptación a la tarea en cuestión. Se han introducido nuevas técnicas que, comparadas con las existentes en el estado del arte, se muestran competitivas y en algunos casos mejores. Lo más importante son las nuevas posibilidades que se han abierto en el procesado digital de imagen con la aplicación de algoritmos que, en principio, están enfocados más a tareas de reconocimiento de patrones y, por ello, de más alto nivel.

En el caso de la detección de bordes se puede decir que se ha definido un método general para la detección de bordes con SVM basado en su capacidad de clasificación y su facilidad de entrenamiento y aprendizaje. A lo largo del capítulo correspondiente se han mostrado ejemplos de su funcionamiento y de las capacidades que presenta, además de ofrecer soluciones a los distintos problemas planteados, muchas de ellas novedosas, como el procesado morfológico con SVM. Las medidas de calidad obtenidas permiten afirmar que las detección de bordes con SVM es competitiva comparada con las establecidas del estado del arte, aunque no es el mejor método para todas las imágenes y en todos los contextos.

En el apartado de segmentación en color se ha demostrado que, comparado con otras técnicas, el uso de SVM (combinadas con tablas de búsqueda) puede ofrecer mejores resul-

tados. Para demostrarlo se ha elegido una aplicación concreta como el reconocimiento de señales de tráfico y se han comparado los resultados obtenidos con los distintos métodos de segmentación. La principal ventaja del método presentado sería su facilidad de entrenamiento puesto que sólo implica el marcado de los colores a detectar en imágenes reales. Esta facilidad además permitirá la generación de diferentes entrenamientos en diferentes escenarios sin una pérdida de tiempo excesiva.

Aunque no es posible identificar un método de segmentación en color que sea el mejor en todos los casos, la principal conclusión a la que se puede llegar es que la umbralización en espacios de color que incorporen una normalización es una buena elección. Además, el uso de tablas de búsqueda con alguna pérdida de información mejora la velocidad e implica que métodos, que de otra manera serían lentos, se pueden usar con buenos resultados. La descomposición acromática es una buena elección ya que muchos de los problemas detectados al clasificar el color vienen de procesar los píxeles acromáticos, que tienden a la inestabilidad. Además, la identificación de señales de tráfico con contenido blanco se mejora usando esta descomposición.

En el caso de la eliminación de ruido queda demostrado que el uso de clasificadores basados en métodos de aprendizaje como las SVM o las RBF pueden mejorar la calidad de reconstrucción de las imágenes contaminadas con ruido impulsivo. Se ha visto qué métodos son más adecuados para cada tasa de ruido, con lo que haciendo uso de una mezcla de varios se podrían realizar mejoras aún mayores. Queda abierta la posibilidad de utilizar la capacidad de detección de ruido para decidir el mejor método a utilizar. Una ventaja de los métodos presentados, que es difícil de evaluar, es su flexibilidad. Puesto que se basan en un entrenamiento artificial parece que modificar ese entrenamiento podría modificar también el comportamiento de la reducción de ruido. Intentar adaptar el método de reducción al tipo de imagen sería el siguiente paso.

## 6.2. Contribuciones originales

### 6.2.1. Detección de bordes

En el capítulo de detección de bordes se presenta un método nuevo basado en la clasificación de píxeles mediante el uso de SVM. En ese contexto, ya de por sí novedoso, se incorporan además las siguientes aportaciones originales:

- Generación de imágenes sintéticas de entrenamiento para la detección de los bordes. Estas imágenes, aunque sintéticas, permiten generalizar el entrenamiento a imágenes reales y evitan el problema de usar modelos basados en imágenes reales que no permiten dicha generalización. Además su generación es configurable y permite adaptarse a contextos más concretos.

- Se introducen varios métodos para la obtención del gradiente de bordes a partir de los valores de clasificación de las SVM para, de esa manera, poder mejorar la detección de bordes.

- Se ha introducido un método para el procesado morfológico basado en SVM que permite gran flexibilidad en la definición de reglas morfológicas porque se hace de manera visual.

Además, se han realizado pruebas para la comparación de los distintos métodos propuestos y su comportamiento en presencia de ruido indicando que el entrenamiento sintético elegido permite "insensibilizar" el detector y con ello reducir el número de falsos bordes detectados.

### 6.2.2. Segmentación de imágenes en color

En el capítulo de segmentación en color la principal novedad es la propuesta de un método basado en las SVM para obtener los píxeles de un determinado color en una imagen. Una vez realizada esta propuesta y en los estudios posteriores se han realizado las siguientes aportaciones:

- Se ha experimentado con distintas posibilidades dentro de las SVM incluyendo la multiclasificación y el uso de diferentes kernel.
- Se ha realizado un estudio del comportamiento del método de segmentación en color con SVM en distintos espacios de color, desde el punto de vista de la calidad y del coste computacional.
- La comparación de la segmentación con SVM se ha realizado basándose en los resultados de su aplicación en un sistema de reconocimiento de señales de tráfico. Es una medida objetiva y que permite determinar los mejores métodos de segmentación para una aplicación determinada.
- En el marco de la comparación de métodos se ha generalizado el uso de la descomposición acromática, que si bien no es una aportación de esta tesis, si lo es el aplicarlo a diferentes espacios de color aparte del RGB.

### 6.2.3. Eliminación de ruido impulsivo

La aportación fundamental de este capítulo es la definición de un método para la eliminación de ruido impulsivo en imágenes mediante el uso de clasificadores (SVM, RBF) para la detección de los píxeles ruidosos y el uso de varias técnicas de recuperación que van desde la mediana, y ciertas modificaciones de la misma, hasta el uso de regresión con SVM. Además se ha realizado las siguientes aportaciones:

- La introducción de imágenes sintéticas tanto para el entrenamiento de clasificación como el de regresión. El diseño de dichas imágenes ha permitido elegir convenientemente los tipos de entrenamiento y que éste sea válido para imágenes reales.
- Se ha realizado un estudio exhaustivo del efecto de los distintos parámetros de entrenamiento (tamaño de las imágenes, ruido añadido, kernel usado, etc) sobre el resultado final de detección.
- Se han diseñado distintos métodos partiendo de una misma idea común, clasificar los píxeles en ruidosos y no ruidosos, y recuperarlos de distintas formas. Dentro de los métodos de recuperación está la mediana, que ofrece buenos resultados directamente, pero en esta tesis se proponen, además, modificaciones de la misma en función de los datos de clasificación. Yendo un paso más allá, se propone para la recuperación,

el uso de la regresión con SVM a partir de un entrenamiento sintético, específico para esta tarea.

- Todos los métodos propuestos, así como varios de la literatura, se comparan tomando distintas medidas de calidad. Esta comparación se realiza tanto de manera numérica, mediante tablas, como usando gráficas que facilitan la comparación.

## 6.3. Futuras líneas de investigación

Los estudios realizados y la implementación de algoritmos necesarios para las distintas aplicaciones dejan abierta la posibilidad de abrir o ampliar distintas líneas de investigación:

- Respecto a la detección de bordes queda abierta la posibilidad de incluir más información en el vector que permite decidir si un determinado punto es un borde o no. Se puede incluir información de texturas, colores, estadísticos de la imagen, etc. Queda entonces abierto un posible estudio sobre qué características de la imagen pueden mejorar la calidad de la detección de bordes.

- Aunque se han realizado un buen número de trabajos sobre las métricas que permiten medir la calidad de los detectores de bordes, ninguna es totalmente independiente de una cierta subjetividad. Siguiendo el ejemplo del estudio realizado en el capítulo de segmentación en color podría investigarse la posibilidad de definir una métrica relacionada con el uso final del detector de bordes estudiado e intentar generalizarla a diversas aplicaciones.

- En el apartado de segmentación de imágenes en color queda por explorar la posibilidad de incluir información no de color sino estructural, por ejemplo, la inclusión de bordes detectados previamente o datos de histograma local o similares. Un estudio en profundidad de posibles características a utilizar sería muy útil para mejorar los resultados. En ese sentido se han realizado estudios preliminares sobre la inclusión de información de color en un entorno mayor al utilizado en esta tesis, de $3 \times 3$, con resultados prometedores.

- Una posibilidad que también podría resultar interesante sería el estudio para la realización de un kernel específico para la clasificación de los colores mediante una métrica basada en la percepción humana. El punto de partida podrían ser los espacios de color CIE-Lab y CIE-Luv que, precisamente, definen distancias de percepción.

- En la detección de ruido, queda la posibilidad de mejorar las imágenes de entrenamiento sintético en el sentido de ampliar la escala de grises o incluir bordes en distintas orientaciones. De esta manera se podría realizar un interesante estudio sobre el efecto del entrenamiento en la calidad final.

- Otra posibilidad es intentar generalizar el método de eliminación de ruido impulsivo presentado en esta tesis a otros tipos de ruido y, especialmente, al ruido impulsivo de amplitud aleatoria y al ruido gaussiano.

- Basándose en la experiencia obtenida en esta tesis se propone también la ampliación de la técnica de reducción de ruido a imágenes en color. Esto supone rediseñar el entrenamiento para reflejar la información de color. Para ello, y vista la literatura al respecto, se deberían explorar distintos espacios de color sobre los que trabajar, especialmente aquellos que separan la información de color de la de iluminación.

- El procesado morfológico presentado en esta tesis se ha utilizado únicamente en una tarea asociada a la eliminación de ruido. Sin embargo, la posibilidad de crear reglas morfológicas complejas mediante un entrenamiento visual puede permitir ampliar las posibilidades de este método. Este podría ser también un nuevo marco de investigación.

# Bibliografía


[Abdou79] Abdou, I. E. y Pratt, W. K. Quantitative design and evaluation of enhancement/thresholding edge detectors. *Proceedings of the IEEE*, 67(5):753–763, 1979. doi:10.1109/PROC.1979.11325.

[Abreu95] Abreu, E. y Mitra, S. K. A signal-dependent rank ordered mean (SD-ROM) filter-a new approach for removal of impulses from highly corrupted images. En *Proc. International Conference on Acoustics, Speech, and Signal Processing ICASSP-95*, tomo 4, páginas 2371–2374. 9–12 Mayo 1995. doi:10.1109/ICASSP.1995.479969.

[Abreu96] Abreu, E., et al. A new efficient approach for the removal of impulse noise from highly corrupted images. *IEEE Transactions on Image Processing*, 5(6):1012–1025, Junio 1996. doi:10.1109/83.503916.

[Akkoul10] Akkoul, S., et al. A new adaptive switching median filter. *Signal Processing Letters, IEEE*, 17(6):587 –590, Junio 2010. ISSN 1070-9908. doi:10.1109/LSP.2010.2048646.

[Alamri10] Alamri, S. S., Kalyankar, N. V., y Khamitkar, S. D. A comparative study of removal noise from remote sensing image. *International Journal of Computer Science Issues*, 7(1):32–36, Enero 2010.

[Anderson72a] Anderson, J. A. Separate sample logistic discrimination. *Biometrika*, 59(1):19–35, abril 1972. doi:10.1093/biomet/59.1.19.

[Anderson72b] —. A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14(3-4):197 – 220, 1972. ISSN 0025-5564. doi:DOI:10.1016/0025-5564(72)90075-2.

[Aoyagi96] Aoyagi, Y. y Asakura, T. A study on traffic sign recognition in scene image using genetic algorithms and neural networks. En *Proc. of the 22nd. IEEE Int. Conf. Industrial Electronics, Control and Instrumentation*, tomo 3, páginas 1838–1843. Taipeh, Taiwan, Agosto 1996.

[Astola97] Astola, J. y Kuosmanen, P. *Fundamentals of nonlinear digital filtering*. Electronic engineering systems series. CRC Press, 1997.

[Bergholm87] Bergholm, F. Edge focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):726–741, 1987.

[Bishop96] Bishop, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1 edición, Enero 1996. ISBN 0198538642.

[Blanz96] Blanz, V., et al. Comparison of view-based object recognition algorithms using realistic 3d models. En *ICANN 96: Proceedings of the 1996 International Conference on Artificial Neural Networks*, páginas 251–256. Springer-Verlag, London, UK, 1996. ISBN 3-540-61510-5.

[Boser92] Boser, B. E., Guyon, I. M., y Vapnik, V. N. A training algorithm for optimal margin classifiers. En *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, páginas 144–152. ACM, New York, NY, USA, 1992. ISBN 0-89791-497-X. doi:10.1145/130385.130401.

[Bowyer98] Bowyer, K. y Dougherty, S. *Emperical Evaluation Techniques in Computer Vision*, capítulo Objective Evaluation of Edge Detectors Using a Formally Defined Framework, páginas 211–228. IEEE Comput. Soc. Press, 1998.

[Bravo07] Bravo, A., Vera, M., y Medina, R. Edge detection in ventriculograms using support vector machine classifiers and deformable models. En Rueda, L., Mery, D., y Kittler, J., editores, *Progress in Pattern Recognition, Image Analysis and Applications*, tomo 4756 de *Lecture Notes in Computer Science*, páginas 793–802. Springer Berlin / Heidelberg, 2007. doi:10.1007/978-3-540-76725-1_82.

[Broomhead88] Broomhead, D. S. y Lowe, D. Multi-variable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[Burges98] Burges, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Busin04] Busin, L., et al. Color space selection for unsupervised color image segmentation by histogram multi-thresholding. En *Proc. International Conference on Image Processing ICIP '04*, tomo 1, páginas 203–206. 24–27 Oct. 2004. doi:10.1109/ICIP.2004.1418725.

[Canny86] Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[Carron94] Carron, T. y Lambert, P. Color edge detector using jointly hue, saturation and intensity. En *Proc. ICIP-94. IEEE International Conference Image Processing*, tomo 3, páginas 977–981. 13–16 Nov. 1994. doi:10.1109/ICIP.1994.413699.

[Chabrier04a] Chabrier, S., et al. A comparative study of supervised evaluation criteria for image segmentation. En *Proceedings of the conference*, páginas 1143–1146. EUSIPCO, 2004.

[Chabrier04b] —. Unsupervised evaluation of image segmentation application to multi-spectral images. En *Proc. 17th International Conference on Pattern Recognition ICPR 2004*, tomo 1, páginas 576–579. 23–26 Aug. 2004. doi:10.1109/ICPR.2004.1334206.

[Chang01] Chang, C.-C. y Lin, C.-J. LIBSVM: a library for support vector machines. Informe técnico, University of Taiwan, 2001. Disponible en: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[Chang08] Chang, C.-C., Hsiao, J.-Y., y Hsieh, C.-P. An adaptive median filter for image denoising. En *Proc. Second International Symposium on Intelligent Information Technology Application IITA '08*, tomo 2, páginas 346–350. 20–22 Dec. 2008. doi: 10.1109/IITA.2008.259.

[Chen99] Chen, T., Ma, K.-K., y Chen, L.-H. Tri-state median filter for image denoising. *IEEE Transactions on Image Processing*, 8(12):1834–1838, Dec. 1999. doi:10.1109/83.806630.

[Chen01] Chen, T. y Wu, H. R. Adaptive impulse detection using center-weighted median filters. *IEEE Signal Processing Letters*, 8(1):1–3, Jan. 2001. doi:10.1109/97.889633.

[Chen06] Chen, Y. W. y Lin, C. J. *Feature extraction, foundations and applications.*, capítulo Combining SVMs with various feature selection strategies. Springer, 2006.

[Cheng01] Cheng, H., et al. Color image segmentation: advances and prospects. *Pattern Recognition*, 34(6):2259–2281, 2001. doi: 10.1016/S0031-3203(00)00149-7.

[Cortes95] Cortes, C. y Vapnik, V. Support-vector networks. *Machine Learning*, 20:273–297, Septiembre 1995. ISSN 0885-6125.

[Cychosz94] Cychosz, J. M. *Efficient binary image thinning using neighborhood maps*, páginas 465–473. Academic Press Professional, Inc., San Diego, CA, USA, 1994. ISBN 0-12-336155-9.

[delaEscalera04] de la Escalera, A., et al. Visual sign information extraction and identification by deformable models for intelligent vehicles. *IEEE Trans. on Intelligent Transportation Systems*, 15(2):57–68, Junio 2004.

[Dougherty98] Dougherty, S. y Bowyer, K. *Empirical Evaluation Techniques in Computer Vision*, capítulo Objective evaluation of edge detectors

using a formally defined framework, páginas 211–234. IEEE Computer Society Press, 1998.

[Farbiz00] Farbiz, F., et al. A new fuzzy logic filter for image enhancement. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(1):110–119, Feb. 2000. doi:10.1109/3477.826951.

[Garcia-Garrido06] Garcia-Garrido, M., Sotelo, M., y Martin-Gorostiza, E. Fast traffic sign detection and recognition under changing lighting conditions. En Sotelo, M., editor, *Proc. IEEE Intelligent Transportation Systems Conference ITSC '06*, páginas 811–816. 2006. doi: 10.1109/ITSC.2006.1706843.

[Gauch92] Gauch, J. M. y Hsia, C. W. Comparison of three-color image segmentation algorithms in four color spaces. En Maragos, P., editor, *Proc. SPIE Visual Communications and Image Processing '92*, tomo 1818 de *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, páginas 1168–1181. Nov 1992. doi:10.1117/12.131388.

[Gil-Jiménez08] Gil-Jiménez, P., et al. Traffic sign shape classification and localization based on the normalized FFT of the signature of blobs and 2D homographies. *Signal Processing*, 88(12):2943–2955, Diciembre 2008. doi:10.1016/j.sigpro.2008.06.019.

[Gómez-Moreno01a] Gómez-Moreno, H., Maldonado-Bascón, S., y López-Ferreras, F. Edge detection in noisy images using the support vector machines. En Mira, J. y Prieto, A., editores, *IWANN (1)*, tomo 2084 de *Lecture Notes in Computer Science*, páginas 685–692. Springer, 2001. ISBN 3-540-42235-8.

[Gómez-Moreno01b] Gómez-Moreno, H., et al. Edge detection by using the support vector machines. En *Proc. ECCTD 01 - European Conference on Circuit Theory and Design*, tomo 3, páginas 145–148. Espoo, Finland, Agosto 28-31 2001.

[Gómez-Moreno01c] —. Extracting illumination from images by using the wavelet transform. En *Proc. IEEE International Conference on Image Processing.*, tomo 2, páginas 265–268. IEEE International Conference on Image Processing. ICIP., Tesalónica, Grecia, Octubre 2001. doi:10.1109/ICIP.2001.958475. Poster.

[Gómez-Moreno02] —. A new and improved edge detector using the support vector machines. En Mastorakis, N., editor, *Proc. WSEAS Int.Conf. on Signal Processing, Robotics and Automation (ISPRA '02)*, páginas 239–243. 2002.

[Gómez-Moreno03] —. Removal of impulse noise in images by means of the use of support vector machines. En *IWANN '03: Proceedings of the 7th*

*International Work-Conference on Artificial and Natural Neural Networks*, páginas 536–543. Springer-Verlag, Berlin, Heidelberg, 2003. doi:http://dx.doi.org/10.1007/3-540-44869-1_68. ISBN: 978-3-540-40211-4.

[Gómez-Moreno10] —. Goal evaluation of segmentation algorithms for traffic sign recognition. *Intelligent Transportation Systems, IEEE Transactions on*, 11(4):917 –930, Dec. 2010. ISSN 1524-9050. doi: 10.1109/TITS.2010.2054084.

[González93] González, R. y Woods, R. *Digital Image proccesing*. Addison-Wesley, 1993.

[Grossberg75] Grossberg, S. A neural model of attention, reinforcement and discrimination learning. *International review of neurobiology*, 18:263–327, 1975. ISSN 0074-7742.

[Grossberg76a] —. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23(3):121–134, julio 1976. ISSN 0340-1200.

[Grossberg76b] —. Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions. *Biological Cybernetics*, 23(4):187–202, agosto 1976. ISSN 0340-1200.

[Gunn98] Gunn, S. R. Support vector machines for classification and regression. Informe técnico, Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, Mayo 1998.

[Han97] Han, W.-Y. y Lin, J.-C. Minimum-maximum exclusive mean (MMEM) filter to remove impulse noise from highly corrupted images. *Electronics Letters*, 33(2):124–125, 16 Jan. 1997.

[Hasan00] Hasan, M. y Marvasti, E. Efficient rank-ordered mean (ROM) techniques for the recovery of isolated losses in highly corrupted images. *IEEE Communications Letters*, 4(10):321–322, Oct. 2000. doi:10.1109/4234.880822.

[Haykin98] Haykin, S. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 2 edición, Julio 1998. ISBN 0132733501.

[Heath97] Heath, M., et al. A robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1338–1359, 1997. ISSN 0162-8828. doi:10.1109/34.643893.

[Hsu03] Hsu, C. W., Chang, C. C., y Lin, C. J. A practical guide to support vector classification. Informe técnico, University of Taiwan, Taipei, 2003.

[Jain89] Jain, A. K. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[Janssen93] Janssen, R., et al. Hybrid approach for traffic sign recognition. En *IEEE International Conference on Intelligent Vehicles*, páginas 390–397. IEEE, 1993.

[Jantzen98] Jantzen, J. Tutorial on fuzzy logic. Informe técnico, Technical University of Denmark, Department of Automation, Lyngby, DENMARK, Agosto 1998.

[Joachims02] Joachims, T. *Learning to Classify Text Using Support Vector Machines : Methods, Theory and Algorithms (The Kluwer International Series in Engineering and Computer Science).* Springer, Abril 2002. ISBN 079237679X.

[Kamada90] Kamada, H., Naoi, S., y Gotoh, T. A compact navigation system using image processing and fuzzy control. En *IEEE Proceedings of Southeastcon*, tomo 1, páginas 337–342. New Orleans, Abril 1990.

[Kim96a] Kim, K. M., et al. Color image quantization using weighted distortion measure of HVS color activity. En *Proc. International Conference on Image Processing*, tomo 3, páginas 1035–1039. 16–19 Sept. 1996. doi:10.1109/ICIP.1996.561015.

[Kim96b] Kim, W.-S. y Park, R.-H. Color image palette construction based on the HSI color system for minimizing the reconstruction error. En *Proc. International Conference on Image Processing*, tomo 3, páginas 1041–1044. 16–19 Sept. 1996. doi: 10.1109/ICIP.1996.561017.

[Kim02] Kim, K. I., et al. Support vector machines for texture classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(11):1542–1550, 2002. ISSN 0162-8828. doi:10.1109/TPAMI.2002.1046177.

[Ko91] Ko, S. J. y Lee, Y. H. Center weighted median filters and their applications to image enhancement. *IEEE Transactions on Circuits and Systems*, 38(9):984–993, Sept. 1991. doi:10.1109/31.83870.

[Kohonen72] Kohonen, T. Correlation matrix memories. *Computers, IEEE Transactions on*, C-21(4):353 –359, 1972. ISSN 0018-9340. doi: 10.1109/TC.1972.5008975.

[Kohonen76] Kohonen, T. y Oja, E. Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics*, 21(2):85–95, junio 1976. ISSN 0340-1200. doi:10.1007/BF01259390.

[Kohonen81] Kohonen, T., Oja, E., y Lehtio, P. *Parallel models of associative memory.*, capítulo Storage and processing of Information in distributed associative memory systems., páginas 49–81. Earlbaum, 1981.

[Kumar02] Kumar, P., Sengupta, K., y Lee, A. A comparative study of different color spaces for foreground and shadow detection for traffic monitoring system. En *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, páginas 100–105. 2002. doi:10.1109/ITSC.2002.1041196.

[Lafuente-Arroyo04] Lafuente-Arroyo, S., et al. Traffic sign classification invariant to rotations using support vector machines. En *Proc. of Advanced Concepts for Intelligent Vision Systems*. Brussels, Belgium, Agosto-Septiembre 2004.

[Lafuente-Arroyo06] —. Road sign tracking with a predictive filter solution. En *IEEE 32nd Annual Conference on Industrial Electronics, IECON 2006*, páginas 3314–3319. Noviembre 2006.

[Lightstone95] Lightstone, M., et al. State-conditioned rank-ordered filtering for removing impulse noise in images. En *Proc. IEEE International Symposium on Circuits and Systems ISCAS '95*, tomo 2, páginas 957–960. 28 Abril–3 Mayo 1995. doi:10.1109/ISCAS.1995.519924.

[Littmann97] Littmann, E. y Ritter, H. Adaptive color segmentation-a comparison of neural and statistical methods. *IEEE Transactions on Neural Networks*, 8(1):175–185, Jan. 1997. doi:10.1109/72.554203.

[Liu02] Liu, H., Liu, D., y Xin, J. Real-time recognition of road traffic sign in motion image based on genetic algorithm. En *Proc. of the 1st. Int. Conference on Machine Learning and Cybernetics*, páginas 83–86. Noviembre 2002.

[Lucchese01] Lucchese, L. y Mitra, S. K. Color image segmentation: A state-of-the-art survey. *Proceedings of the Indian National Science Academy (INSA-A)*, 67-A:207–221, 2001.

[Maldonado-Bascón07] Maldonado-Bascón, S., et al. Road-sign detection and recognition based on support vector machines. *IEEE Trans. on Intelligent Transportation Systems*, 8(2):264–278, 2007. doi:10.1109/TITS.2007.895311.

[Marr80] Marr, D. y Hildreth, E. A theory of edge detection. *Proceedings of the Royal Society of London B*, 207:187–217, 1980.

[Mcculloch43] Mcculloch, W. S. y Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysic*, 5:115–133, 1943.

[Mélange11] Mélange, T., Nachtegael, M., y Kerre, E. Fuzzy random impulse noise removal from color image sequences. *Image Processing, IEEE Transactions on*, 20(4):959 –970, Abril 2011. ISSN 1057-7149. doi:10.1109/TIP.2010.2077305.

[Minsky69] Minsky, M. L. y Papert, S. A. *Perceptrons*. The MIT Press, diciembre 1969. ISBN 0262631113.

[Moody89] Moody, J. y Darken, C. J. Fast learning in networks of locally-tuned processing units. *Neural Comput.*, 1(2):281–294, 1989. ISSN 0899-7667. doi:10.1162/neco.1989.1.2.281.

[Ohta80] Ohta, Y., Kanade, T., y Sakai, T. Color information for region segmentation. *Computer Graphics and Image Processing*, 13(3):222 – 241, Julio 1980.

[Otsu79] Otsu, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, Enero 1979.

[Pankajakshan07] Pankajakshan, P. y Kumar, V. Detail-preserving image information restoration guided by SVM based noise mapping. *Digit. Signal Process.*, 17(3):561–577, 2007. ISSN 1051-2004. doi: 10.1016/j.dsp.2006.11.006.

[Pitas90] Pitas, I. y Venetsanopoulos, A. *Nonlinear digital filters*. Kluwer Academic Publishers, Boston, 1990.

[Plataniotis00] Plataniotis, K. N. y Venetsanopoulos, A. N. *Color image processing and applications*. Springer-Verlag New York, Inc., New York, NY, USA, 2000. ISBN 3-540-66953-1.

[Platt99] Platt, J. C. Probabilities for SV machines. En A. Smola, B. S., P. Bartlett y Schuurmans, D., editores, *Advances in Large Margin Classifiers*, páginas 61–74. MIT Press, 1999.

[Pratt01] Pratt, W. K. *Digital image processing*. John Willey and Sons, Inc, New York, 2001.

[Rothwell95] Rothwell, C., et al. Driving vision by topology. En *Proceedings of International Symposium on Computer Vision*. 1995.

[Russo95] Russo, F. y Ramponi, G. A fuzzy operator for the enhancement of blurred and noisy images. *IEEE Transactions on Image Processing*, 4(8):1169–1174, Aug. 1995. doi:10.1109/83.403425.

[Russo96a] —. A fuzzy filter for images corrupted by impulse noise. *IEEE Signal Processing Letters*, 3(6):168–170, Jun 1996. ISSN 1070-9908. doi:10.1109/97.503279.

[Russo96b] —. Removal of impulse noise using a FIRE filter. En *Proc. International Conference on Image Processing*, tomo 1, páginas 975–978. 16–19 Sept. 1996. doi:10.1109/ICIP.1996.561068.

[Russo99] Russo, F. Evolutionary neural fuzzy systems for noise cancellation in image data. *IEEE Transactions on Instrumentation and Measurement*, 48(5):915–920, Oct 1999. ISSN 0018-9456. doi: 10.1109/19.799647.

[Russo00] —. Noise removal from image data using recursive neurofuzzy filters. *IEEE Transactions on Instrumentation and Measurement*, 49(2):307–314, Apr 2000. ISSN 0018-9456. doi:10.1109/19.843069.

[Schulte06a] Schulte, S., et al. A fuzzy impulse noise detection and reduction method. *IEEE Transactions on Image Processing*, 15(5):1153–1162, Mayo 2006. doi:10.1109/TIP.2005.864179.

[Schulte06b] —. Fuzzy two-step filter for impulse noise reduction from color images. *IEEE Transactions on Image Processing*, 15(11):3567–3578, Nov. 2006. doi:10.1109/TIP.2006.877494.

[Schulte07a] Schulte, S., De Witte, V., y Kerre, E. A fuzzy noise reduction method for color images. *IEEE Transactions on Image Processing*, 16(5):1425–1436, Mayo 2007. doi:10.1109/TIP.2007.891807.

[Schulte07b] Schulte, S., et al. A new fuzzy color correlated impulse noise reduction method. *IEEE Transactions on Image Processing*, 16(10):2565–2575, Oct. 2007. doi:10.1109/TIP.2007.904960.

[Shawe-Taylor04] Shawe-Taylor, J. y Cristianini, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972.

[Smith78] Smith, A. R. Color gamut transform pairs. En *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, páginas 12–19. ACM, New York, NY, USA, 1978. doi:http://doi.acm.org/10.1145/800248.807361.

[Souza10] Souza, C. R. Kernel functions for machine learning applications. Informe técnico, 2010. Disponible en: http://crsouza.blogspot.com/2010/03/kernel-functions-for-machine-learning.html.

[Sucher94] Sucher, R. Removal of impulse noise by selective filtering. En *Proc. ICIP-94. IEEE International Conference Image Processing*, tomo 2, páginas 502–506. 13–16 Nov. 1994. doi:10.1109/ICIP.1994.413621.

[Sucher95a] —. A recursive nonlinear filter for removal of impulse noise. En *Proc. International Conference on Image Processing*, tomo 1, páginas 183–186. 23–26 Oct. 1995. doi:10.1109/ICIP.1995.529576.

[Sucher95b] —. A self-organising nonlinear noise filtering scheme. En *Conference Record of the Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, tomo 1, páginas 681–684. 30 Oct.–2 Nov. 1995. doi:10.1109/ACSSC.1995.540636.

[Tao06] Tao, D., et al. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1088 –1099, jul. 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.134.

[Tepichin-Rodriguez95] Tepichin-Rodriguez, E., Suarez-Romero, J. G., y Ramirez Zabaleta, G. Hue, brightness, and saturation manipulation of diffractive colors. *Optical Engineering*, 34(10):2886–2890, 1995. doi: 10.1117/12.210744.

[Toprak07] Toprak, A. y Güler, I. Impulse noise reduction in medical images with the use of switch mode fuzzy adaptive median filter. *Digit. Signal Process.*, 17:711–723, Julio 2007. ISSN 1051-2004. doi: 10.1016/j.dsp.2006.11.008.

[Tsang96] Tsang, P. y Tsang, W. Edge detection on object color. En *Proc. International Conference on Image Processing*, tomo 3, páginas 1049–1052. 16–19 Sept. 1996. doi:10.1109/ICIP.1996.561021.

[Vandenbroucke98] Vandenbroucke, N., Macaire, L., y Postaire, J.-G. Color pixels classification in an hybrid color space. En Macaire, L., editor, *Proc. International Conference on Image Processing ICIP 98*, tomo 1, páginas 176–180. 1998. doi:10.1109/ICIP.1998.723452.

[Vandenbroucke00] —. Color image segmentation by supervised pixel classification in a color texture feature space. application to soccer image segmentation. En *Proc. 15th International Conference on Pattern Recognition*, tomo 3, páginas 621–624. 3–7 Sept. 2000. doi: 10.1109/ICPR.2000.903622.

[Vapnik71] Vapnik, V. y Chervonenkis, A. On the uniform convergence of relative frequencies of events to their probabilities. *Theoretical Probabibility and its Applicactions*, 17:264–280, 1971.

[Vapnik95] Vapnik, V. *The Nature of Statistical Learning Theory.* Springer-Verlog, New York, 1995.

[Vapnik98] —. *Statistical Learning Theory.* John Willey and Sons, Inc, New York, 1998.

[Vertan00] Vertan, C. y Boujemaa, N. Color texture classification by normalized color space representation. En *Proceedings of the International Conference on Pattern Recognition*. Barcelona, 2000.

[Wang04] Wang, Z., et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactios on Image Processing*, 13(4):600–612, Abril 2004. doi:10.1109/TIP.2003.819861.

[Wang07] Wang, W., Liao, H., y Huang, Y. Rock fracture tracing based on image processing and SVM. *International Conference on Natural Computation*, 1:632–635, 2007. doi:10.1109/ICNC.2007.643.

[Xu09] Xu, Z., et al. Geometric features-based filtering for suppression of impulse noise in color images. 18(8):1742–1759, 2009. doi: 10.1109/TIP.2009.2022207.

[Yang95] Yang, R., et al. Optimal weighted median filtering under structural constraints. *IEEE Transactions on Signal Processing*, 43(3):591–604, Marzo 1995. doi:10.1109/78.370615.

[Yin91] Yin, L., Astola, J., y Neuvo, Y. Adaptive neural filters. En *Proc. IEEE Workshop Neural Networks for Signal Processing [1991]*, páginas 503–512. 30 Sept.–1 Oct. 1991. doi:10.1109/NNSP.1991.239491.

[Yin92] —. Neural filters: a class of filters unifying fir and median filters. En *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-92*, tomo 4, páginas 53–56. 23–26 Marzo 1992. doi:10.1109/ICASSP.1992.226413.

[Zadeh65] Zadeh, L. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[Zadeh78] —. Fuzzy sets as a basis for theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[Zhang96] Zhang, Y. J. A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335–1346, Agosto 1996. doi:10.1016/0031-3203(95)00169-7.

[Zhang01a] Zhang, X.-P. Thresholding neural network for adaptive noise reduction. *IEEE Transactions on Neural Networks*, 12(3):567–584, Mayo 2001. doi:10.1109/72.925559.

[Zhang01b] Zhang, Y. J. A review of recent evaluation methods for image segmentation. En *Proc. Symposium on Signal Processing and its Applications, Sixth International 2001*, tomo 1, páginas 148–151. Agosto 2001. doi:10.1109/ISSPA.2001.949797.

[Zheng04] Zheng, S., Liu, J., y Tian, J. W. A new efficient SVM-based edge detection method. *Pattern Recognition Letters*, 25(10):1143 – 1154, 2004. ISSN 0167-8655. doi:10.1016/j.patrec.2004.03.009.

[Zou05] Zou, A.-M., Hou, Z.-G., y Tan, M. Support vector machines (SVM) for color image segmentation with applications to mobile robot localization problems. En Huang, D.-S., Zhang, X.-P., y Huang, G.-B., editores, *Advances in Intelligent Computing*, tomo 3645 de *Lecture Notes in Computer Science*, páginas 443–452. Springer Berlin / Heidelberg, 2005. doi:10.1007/11538356_46.

# Apéndice A

# Publicaciones a las que ha dado lugar la realización de la tesis

En este apéndice se enumeran las publicaciones a las que ha dado lugar la realización de la presente tesis, tanto en revistas como congresos nacionales e internacionales.

## A.1. Revistas

- **H. Gómez-Moreno**, S. Maldonado-Bascón, P. Gil-Jiménez y S. Lafuente-Arroyo. Goal evaluation of segmentation algorithms for traffic sign recognition. *Intelligent Transportation Systems, IEEE Transactions on*, 11(4):917 –930, Diciembre 2010. ISSN 1524-9050.

- S. Maldonado-Bascón, S. Lafuente-Arroyo, P. Gil-Jiménez, **H. Gómez-Moreno** y F. López-Ferreras. Road-sign detection and recognition based on support vector machines. *IEEE Trans. on Intelligent Transportation Systems*, 8(2):264–278, Junio 2007. ISSN 1524-9050. 10.1109/TITS.2007.895311.

## A.2. Congresos internacionales

- Sergio Lafuente-Arroyo and Saturnino Maldonado-Bascón and **Hilario Gómez-Moreno** and Pedro Gil-Jiménez. False alarm filtering in a vision traffic sign recognition system - an approach based on adaboost and heterogeneity of texture. En *Proceedings of the conference*, páginas 269–276. III International Conference on Agents and Artificial Intelligent (ICAART), Roma (Italy), Enero 2011.

- **Hilario Gómez-Moreno**, Saturnino Maldonado-Bascón, Francisco López-Ferreras, y Pedro Gil-Jiménez. Removal of impulse noise in images by means of the use of support vector machines. En José Mira y José R. Álvarez, editores, *IWANN.*, volumen 2687 of *Lecture Notes in Computer Science*, páginas 536–543, Mahón, España., Junio 2003. 7th. International Work Conference on Artificial and Natural Neural Networks, Springer. ISBN 3-540-40211-X.

- **Hilario Gómez-Moreno**, Pedro Gil-Jiménez, Sergio Lafuente-Arroyo, Raúl Vicen-Bueno, y Rocio Sánchez-Montero. *Recent Advances in Intelligent Systems and Signal Processing*, capítulo Color images segmentation using the Support Vector Machines, páginas 151–155. World Scientific and Engineering Society Press, 222 Rosewood Drive, Danvers, MA 01923, USA, 2003.

- **Hilario Gómez-Moreno**, S. Maldonado-Bascón, F. López-Ferreras, y Pedro Gil-Jiménez. *Advances in Systems Engineering, Signal Processing and Communicactions*, capítulo A new and improved edge detector using the Support Vector Machines, páginas 239–243. World Scientific and Engineering Society Press, 222 Rosewood Drive, Danvers, MA 01923, USA, 2002.

- **Hilario Gómez-Moreno**, S. Maldonado-Bascón, F. López-Ferreras, M. Utrilla-Manso, y Pedro Gil-Jiménez. *Advances in Signal Processing and Computer Technologies.*, capítulo A modified median filter for the removal of impulse noise based on the support vector machines., páginas 9–14. World Scientific and Engineering Society Press, 222 Rosewood Drive, Danvers, MA 01923, USA, 2001.

- **Hilario Gómez-Moreno**, Saturnino Maldonado-Bascón, y Francisco López-Ferreras. Edge detection in noisy images using the support vector machines. En José Mira y Alberto Prieto, editores, *IWANN.*, volumen 2084 of *Lecture Notes in Computer Science*, páginas 685–692, Granada, España., Junio 2001. 6th. International Work-Conference on Artificial and Natural Neural Networks, Springer. ISBN 3-540-42235-8.

- **Hilario Gómez-Moreno**, Saturnino Maldonado-Bascón, Francisco López-Ferreras, F. Javier Acevedo Rodríguez y Pilar Martín Martín. Edge detection by using the support vector machines. En Proc. ECCTD 01 - European Conference on Circuit Theory and Design, tomo 3, páginas 145–148. Espoo, Finland, Agosto 28-31 2001.

## A.3. Congresos nacionales

- **Hilario Gómez-Moreno**, Saturnino Maldonado-Bascón, Manuel Utrilla Manso, Pilar Martín Martín. Eliminación de ruido impulsivo en imágenes mediante el uso de máquinas de vectores soporte. En Actas congreso XVI URSI ,Villaviciosa de Odón, España., Junio 2001.

# Índice alfabético