DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

PROYECTO FIN DE CARRERA

CLASIFICACIÓN DE ESCENAS EN CONTENIDO AUDIOVISUAL

Autor: Víctor Suárez Paniagua

Tutor: Iván González Díaz

Leganés, noviembre de 2012

Título: Clasificación de escenas en contenido audiovisual
Autor: Víctor Suárez Paniagua
Director: Iván González Díaz

EL TRIBUNAL

Presidente: Fernando Díaz de María

Vocal: Julio Villena Román

Secretario: Manuel de Frutos López

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 19 de noviembre de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Resumen

En este proyecto se han diseñado algoritmos para la clasificación de escenas mediante categorías genéricas tales como: costa, calle, montaña, cielo, oficina, casa, etc. El objetivo de los algoritmos desarrollados es ser integrado en un sistema de anotación automático de contenidos visuales. Para ello, se realiza un estudio comparativo y sistemático de dos alternativas del estado del arte: el método de extracción de esencia de imágenes, llamado *gist descriptor*, y el método *bag of words* para la extracción de características. Para la clasificación se utiliza, además, una máquina de vector soporte.

La motivación del proyecto es, en primer lugar, desarrollar un sistema que pueda ser capaz de clasificar una gran base de datos de imágenes según el tipo de escena visual, y, en segundo lugar, que el estudiante conozca y sepa utilizar diferentes métodos de clasificación de escenas audiovisuales, pueda evaluarlas y tomar conclusiones.

Palabras clave: clasificación escenas, contenido audiovisual, etiquetado, extracción de características, gist descriptor, bag of words, máquinas de vectores soporte.

Abstract

In this project I had designed algorithms for scene classification using generic categories such as: coast road, mountain, sky, office, home, etc.. The developed algorithms have been integrated into a system for automatic annotation of visual content. For this purpose, I have performed a comparative and systematic study of two state of the art alternatives: the gist extraction method of images, called gist descriptor, and the bag of words method for feature extraction. For the classification, additionaly, I use a support vector machine.

This project aims to, on the one hand, develop a system that will be able to classify a large database of images according to the type of audiovisual scene, and, on the other, that the student learns and knows how to use different methods of classification for visual scenes, as well as he can evaluate and draw his own conclusions from the results.

Keywords: classification of scenes, audiovisual content, labeled, feature extraction, gist descriptor, bag of words, support vector machine.

Índice general

1. Introducción y objetivos 1
1.1 Introducción 1
1.2 Objetivos 3
1.3 Estructura de la memoria 5

2. Estado del arte 6
2.1 Introducción 6
2.2 Anotación de imágenes 7
2.3 Clasificación de imágenes 10
2.3.1 Tipos de clasificación de imágenes 10
2.3.2 Sistemas de reconocimiento de patrones 11
2.4 Métodos de extracción de características 14
2.4.1 Características de bajo nivel 14
2.4.1 *SIFT* 16
2.4.1 *Descriptor Gist* 19
A. Introducción 19
B. Estudios sobre el reconocimiento de escenas 19
C. La forma de la escena. Definición de *Spatial Envelope* 20
D. Atributos a modelar con el *Spatial Envelope* 20
E. Filtrado de Gabor 23

F. Representaciones basadas en imagen 25
G. *Discriminant Spectral Template* 28
H. Modelado de atributos 29
I. *Gist* para clasificación de imágenes 30
2.4.2 *Bag of words* 31
A. Introducción 31
B. Implementación del modelo 31
C. Generación del vocabulario 32
D. Algoritmo *k-means* 34
E. Histograma normalizado 34
2.5 Algoritmos de aprendizaje máquina 35
2.5.1 Support Vector Machine (SVM) 35
2.6 Métodos de evaluación 39
2.6.1 Accuracy 39
2.6.1 Average Precision 39
3. Desarrollo del proyecto 42
3.1 Introducción 42
3.2 Base de datos empleada 43
3.3 Implementación del proyecto 44
3.3.1 Esquema del proyecto 44
3.3.2 Lenguaje de programación 45
3.3.3 Librerías externas 45
A. *Gist Descriptor* 46
B. *VLFeat* 46
C. *LibSVM* 46
D. *LS_SVMlab1.8* 47
3.3.4 Extracción de características 47
A. *Gist Descriptor* 47
B. *Bag of words* 49
3.3.5 Etiquetado de datos 54
3.3.6 Validación del clasificador tipo *SVM* 55
A. *Gist Descriptor* 57
B. *Bag of words* 61

3.3.7 Entrenamiento del clasificador tipo *SVM* 61
3.3.8 Evaluación del clasificador tipo *SVM* 62
3.3.9 Resultados del clasificador tipo *SVM* 62
3.4 Resultados 63
3.4.1 Validación de parámetros 63
A. *Gist Descriptor* 63
B. *Bag of words* 66
3.4.2 Comparativas 66
A. *DST* vs *WDST* y *Lineal* vs *RBF* 66
B. *Gist descriptor* vs *Bag of words* 71
C. Coste computacional 73
D. Resultados visuales 74
E. Conclusiones al estudio 75
3.5 Integración en el sistema 76

4. Conclusiones y líneas futuras 77
4.1 Introducción 77
4.2 Conclusiones 78
4.3 Líneas futuras 80

5. Presupuesto 82
5.1 Introducción 82
5.2 Esquema de las fases del proyecto 83
5.3 Diagrama de Gantt 85
5.4 Presupuesto final 86
5.4.1 Coste material 86
5.4.2 Honorarios 87
5.4.3 Presupuesto total 88

6. Anexo 89
6.1 Introducción 89
6.2 Mecánica del programa 90
6.3 Archivos programados 93
6.3.1 GIST 93
A. Gist.m 93
B. ExtraccionGist.m 94

C. EtiquetadoGist.m 96
D. SVMGistValidation.m 97
E. SVMGistTrain.m 102
F. SVMGistTest.m 103
G. ResultadosGist.m 104
6.3.2 BOW 105
A. BOW.m 105
B. ExtraccionBOW.m 106
C. HistogramaBOW.m 107
D. EtiquetadoBOW.m 108
E. SVMBOWValidation.m 109
F. SVMBOWTrain.m 110
G. SVMBOWTest.m 111
H. ResultadosBOW.m 112
6.3.3 Funciones auxiliares 113
A. NumeroEscenas.m 113
B. ComputeAP.m 113

Índice de figuras

Figura 1. Anotación manual aplicada a la imagen de un animal 8
Figura 2. Anotación manual aplicada a la imagen de un interior 8
Figura 3. Detección de imágenes con etiquetado 9
Figura 4. Segmentación de imágenes 9
Figura 5. Modelo de un reconocedor de patrones 12
Figura 6. Imagen dividida en rejillas de tamaño fijo 15
Figura 7. Detección de puntos clave 15
Figura 8. Representación del proceso de Diferencia de Gaussianas 17
Figura 9. Localización de puntos clave y asignación de la orientación 18
Figura 10. Descriptor SIFT 19
Figura 11. Ejemplos de grado de naturaleza a) Bajo grado b) Alto grado 21
Figura 12. Ejemplos de grado de apertura a) Bajo grado b) Alto grado 21
Figura 13. Ejemplos de grado de aspereza a) Bajo grado b) Alto grado 22
Figura 14. Ejemplos de grado de expansión a) Bajo grado b) Alto grado 22
Figura 15. Ejemplos de grado de rugosidad a) Bajo grado b) Alto grado 23
Figura 16. Filtro de Gabor 24
Figura 17. Ejemplos de Filtros de Gabor aplicadas a imágenes 25
Figura 18. *Bag of words* para una imagen con descriptores *SIFT* 32
Figura 19. Extracción de palabras clave de una imagen 32
Figura 20. Creación de un vocabulario a partir de imágenes 33

Figura 21. Ejemplo de normalización de histograma 34
Figura 22. Ejemplo de los márgenes de *SVM* 36
Figura 23. Ejemplo gráfico del truco del *kernel* 37
Figura 24. Explicación gráfica con diagramas de Venn de *Precision* y *Recall* 40
Figura 25. Ejemplos de la base de datos 43
Figura 26. Esquema del proyecto 44
Figura 27. Ejemplo de extracción de *Gist descriptor DST* 48
Figura 28. Ejemplo de extracción de *Gist descriptor WDST* 48
Figura 29. Ejemplo de extracción de detector *SIFT* 49
Figura 30. Ejemplo de extracción de descriptor *SIFT* 50
Figura 31. Ejemplo de agrupamiento de los datos en un plano con un número de centroides igual a 3 51
Figura 32. Ejemplo de agregación de los datos en un plano con un número de centroides igual a 3 52
Figura 33. Ejemplo del proyecto de histograma 53
Figura 34. Ejemplo del proyecto de histograma normalizado 53
Figura 35. Explicación de etiquetado de datos 54
Figura 36. Esquema de validación cruzada con $k = 4$ 55
Figura 37. Modo de búsqueda para el parámetro *C* máximo 58
Figura 38. Modo de búsqueda para el parámetro *C* mínimo 58
Figura 39. Modo de búsqueda para el parámetro *C* máximo 60
Figura 40. Modo de búsqueda para el parámetro *C* mínimo 60
Figura 41. Modo de búsqueda para la variable *gamma* máximo y *gamma* mínimo respectivamente 60
Figura 42. Modos de búsqueda para el parámetro *C* y la variable *gamma* 60
Figura 43. Ejemplo gráfico de la validación para los parámetros *C* y *gamma* 65
Figura 44. Comparativa *DST* vs *WDST* para una imagen de 'MITstreet' 68
Figura 45. Comparativa *DST* vs *WDST* para una imagen de 'MITforest' 69
Figura 46. Resultados visuales para *DST* con *SVM* Lineal en 'MITmountain' 74
Figura 47. Resultados visuales para *WDST* con *SVM RBF* en 'MIThighway' 74
Figura 48. Resultados visuales para *BoW* en 'kitchen' 74
Figura 49. Ejemplo de *Spatial Pyramid Matching* 80
Figura 50. Diagrama de Gantt del proyecto 85
Figura 51. Coste de dirección 87

Figura 52. Coste de realización 88
Figura 53. Ejemplo de ordenación de la base de imágenes 91
Figura 54. Directorio del programa 91
Figura 55. Subdirectorios del programa 91

Índice de ecuaciones

Ecuación 1. Función compleja que conforma el filtro de Gabor 24
Ecuación 2. Pre filtrado de la intensidad de la imagen 25
Ecuación 3. *DFT* para una imagen 26
Ecuación 4. *WFT* para una imagen 26
Ecuación 5. Energía espectral 26
Ecuación 6. Espectrograma 26
Ecuación 7. Descomposición de la energía espectral 27
Ecuación 8. Descomposición del espectrograma 27
Ecuación 9. Estimación para la energía espectral 28
Ecuación 10. *Discriminant Spectral Template* 28
Ecuación 11. Estimación para la espectrograma 28
Ecuación 12. *Windowed Discriminant Spectral Template* 28
Ecuación 13. Componentes del vector *DST* 30
Ecuación 14. Componentes del vector *WDST* 30
Ecuación 15. Hiperplano óptimo *SVM* 36
Ecuación 16. Función de coste *SVM* 36
Ecuación 17. Ecuación de *Accuracy* en la clasificación binaria 39
Ecuación 18. *Precision* 40
Ecuación 19. *Recall* 40
Ecuación 20. *Average Precision* 41

Índice de tablas

Tabla 1. Resultado de validación para *DST* con *SVM* lineal 63
Tabla 2. Resultado de validación para *DST* con *SVM RBF* 64
Tabla 3. Resultado de validación para *WDST* con *SVM* lineal 64
Tabla 4. Resultado de validación para *WDST* con *SVM RBF* 65
Tabla 5. Resultado de validación para *BoW* 66
Tabla 6. Resultados finales para *Gist descriptor* de la clasificación de *test* 67
Tabla 7. Medias para cada categoría de los resultados finales en *Gist* 69
Tabla 8. Resultados finales para *BoW* de la clasificación de *test* 70
Tabla 9. Medias para cada categoría de los resultados finales en *BoW* 71
Tabla 8. Tabla de las fases del proyecto y sus correspondientes tareas 84

Capítulo 1

Introducción y objetivos

1.1 Introducción

Este capítulo del proyecto pretende dar una introducción e idea general de lo que se va a tratar en la memoria, exponiendo los principales objetivos y las fases del desarrollo que se han ido siguiendo para completar dichos objetivos. Además, para facilitar la lectura, se realizará un resumen sobre la estructura de la memoria en la que se incluirá una breve descripción de cada capítulo.

En este documento se presenta un proyecto de fin de carrera cuyo objetivo principal es crear una aplicación que permita clasificar imágenes en distintos tipos de categorías visuales según la escena que se este mostrando. Para ello, se ha escogido una base de datos grande y con diferentes categorías para evaluar sus características, estudiar las diferentes maneras de extracción de su contenido visual y realizar los modelos de clasificación. Dichos modelos se integrarán después en un sistema de anotación

automática de contenido audiovisual, para ser posible su etiquetado en las categorías mencionadas.

La clasificación de imágenes surge de la necesidad de sistemas de gestión automatizada que sustituyan a la gestión manual, la cual conlleva el etiquetado imagen a imagen por parte del usuario. Es un campo en continua investigación, ya que su utilidad cobra gran importancia en ámbitos donde se tiene una gran base de datos de imágenes con una cierta dispersión, y se requiere una cierta ordenación de los datos. Entre sus usos comerciales destacan:

- los buscadores de imágenes, en los que se intenta encontrar una imagen a partir de una consulta basándose en elementos constituyentes de la imagen y no en anotaciones textuales
- la detección de rostros y rasgos faciales, en los que se pueda discernir el estado de ánimo de una persona (risa, llanto, enfado…)
- la auto descripción de imágenes para personas con discapacidad visual.

En sus usos domésticos la clasificación de imágenes ha cobrado gran importancia por dos motivos:

- auge de las cámaras digitales, lo que conlleva a los usuarios almacenar gran cantidad de imágenes sin clasificar,
- y la aparición de blogs y redes sociales, en las que los usuarios desean clasificar las imágenes según su contenido, y, además, poder hacer búsquedas entre ellas [34].

En primera instancia, se buscan métodos de extracción de características de la imagen que hagan posible su descripción en base a su contenido. En este proyecto hemos elegido estudiar, analizar y comparar los resultados de las formas de extracción de características: el *gist descriptor* de Oliva et Torralba [2], y el *bag of words* (*BoW*) [3], que en base a sus resultados con respecto a nuestra base de datos seleccionaremos la que mejor se adecúe a nuestras categorías para ser integrada en nuestro sistema. Después de ello, debemos utilizar una Máquina Vector Soporte (*SVM*) [22] para hacer posible su clasificación. Todo esto será empleado en el software matemático *MATLAB*.

1.2 Objetivos

El objetivo fundamental del proyecto se centra en extraer modelos eficientes para la clasificación de imágenes en un cierto número de categorías. Para ello, se debe realizar un estudio extenso sobre las formas de extracción de características que hemos escogido, en el que tendremos que verificar cuál de ellas se ajusta más a la clasificación óptima para nuestras categorías. Después, estos modelos serán integrados en un sistema de anotación automática de contenidos que deberá distinguir las imágenes en el número de categorías seleccionadas. También, el sistema deberá ser capaz de realizar una clasificación binaria, ya que varias imágenes pueden contener más de una categoría a la vez. Adicionalmente, se pretende que el coste computacional no sea muy elevado, si bien no se requiere tiempo real.

Cabe recalcar que el proyecto no se centra en clasificar imágenes de objetos, sino que se orienta a la clasificación de imágenes que contengan escenas visuales del mundo real. Una aproximación a la definición de escena la consideraremos con una relativa distancia entre el observador y el plano fijado en la escena, en contraposición a un objeto o una textura. Por tanto, mientras una imagen de un objeto se sitúa en una distancia cercana (de 1 a 2 metros), una escena es una representación de un lugar en el cuál nosotros nos podríamos mover físicamente (normalmente más de 5 metros de distancia al observador) [2].

Para evaluar nuestros sistemas, se ha empleado las categorías seleccionadas pertenecen a la base de datos *Fifteen Scene Categories* de *The Ponce Group* [9], que tiene un total de 4.485 imágenes, y se compone de quince categorías diferentes tales como: costa, autopista, bosque, edificios, oficina, habitaciones, etc.

Además, el proyecto trata de cumplir los siguientes objetivos particulares:

- Implementar una solución de clasificación de escenas que utilice el *Gist descriptor*.

- Analizar las posibles parametrizaciones del descriptor *Gist* y elegir la más adecuada para nuestro problema.

- Implementar una solución de clasificación de escenas que utilice el modelo *Bag of words*. En este caso, no se analizarán posibles configuraciones sino que se utilizarán las recomendaciones dadas en [6].

- Realizar un estudio comparativo para ambas alternativas atendiendo a factores relevantes como la calidad de la clasificación y el coste computacional.

- Seleccionar la alternativa más adecuada para el escenario propuesto.

1.3 Estructura de la memoria

La memoria se estructura en:

- **Capítulo 2:** se hará una introducción a los conceptos que toman relevancia en el proyecto, tales como, anotación de imágenes y sistemas de clasificación de imágenes. También se tratarán las técnicas involucradas en su realización como son los métodos de extracción de características, técnicas de agrupamiento, algoritmos de aprendizaje máquina para la clasificación y el método utilizado para evaluar el resultado final.

- **Capítulo 3:** describirá el desarrollo e implementación del proyecto, comentando los elementos que toman parte en él (bases de datos, librerías externas…) y cómo se han implementado. Además, se explicará los experimentos que se han realizado, la solución adoptada, la forma de implementar el modelo funcional, y la evaluación final del proyecto.

- **Capítulo 4:** presentará las conclusiones obtenidas del proyecto y las líneas futuras que permitirán extenderlo.

- **Capítulo 5:** se realizará un diagrama de Gantt de las diferentes fases del proyecto e incluirá el presupuesto económico de la realización del proyecto, detallando costes materiales, honorarios…

- **Capítulo 6:** incluirá la librería de funciones programada para la realización del proyecto, la forma de utilización del programa y las funciones de librerías externas que se han utilizado.

Capítulo 2

Estado del arte

2.1 Introducción

Este capítulo hace referencia a la base teórica sobre la que se sustenta el proyecto. Se hará un resumen de los precedentes de las técnicas que han sido utilizadas, y se repasarán comentando su funcionamiento desde el punto de vista teórico. Primeramente, se describirán los métodos de anotación de imágenes y los algoritmos que se utilizan en la clasificación de imágenes. Seguidamente, se comentarán los métodos de extracción de características que han sido utilizadas en el proyecto, el método *gist descriptor* y el método *bag of words*. Después, se hará una introducción a los algoritmos de aprendizaje máquina haciendo hincapié en las máquinas de vector soporte, pues han sido la alternativa utilizada en este proyecto. Por último, se hablará de los métodos de evaluación que se han escogido para definir la solución apropiada para nuestro problema de clasificación.

2.2 Anotación de imágenes

La anotación automática de imágenes es el proceso mediante el cual un ordenador asigna automáticamente palabras clave, relacionadas con el contenido visual de la imagen. Estas técnicas son utilizadas en visión artificial, principalmente para organizar y localizar imágenes en una base de datos. Pueden considerarse como un tipo de clasificación de imágenes con un número muy grande de múltiples clases. Habitualmente, el análisis de imagen mediante la extracción de características y las palabras entrenadas para la anotación, se utilizan por técnicas de aprendizaje de máquina para intentar aplicar automáticamente anotaciones a las imágenes nuevas.

Los sistemas *CBIR* resuelven el problema de búsqueda de imágenes digitales en grandes bases de datos, haciendo uso de los contenidos de las imágenes en sí mismas, más que confiar en la información textual que las rodea. Estas técnicas usan las características extraídas automáticamente a partir del contenido (color, textura y forma) como criterios de búsqueda [36]. Las ventajas de la anotación automática de la imagen en comparación con la recuperación de imágenes basada en contenido (*CBIR*) son que las consultas pueden ser especificadas por el usuario [35]. En ocasiones, ambas técnicas pueden ser compatibles para llegar a mejores resultados.

Para encontrar la similaridad entre el texto y la imagen, cada porción de ésta debe estar descrita por unas palabras que puedan explicar su contenido al completo. Asignando palabras clave a las porciones de imagen podremos tener una idea de la imagen en general [10]. Por tanto, cada una de ellas podrá ser representada por un conjunto de palabras relacionadas con las señales visuales. Es posible que esta misma señal visual pueda estar compartida por más de una imagen. Por esta razón, si alguna de las señales visuales son las mismas, deberían pertenecer a la misma categoría. Por ello, para determinar esta similaridad entre texto e imagen, se necesita agrupar las señales visuales similares para construir señales patrón y analizar la similitud entre texto e imagen.

Figura 1: "Anotación manual aplicada a la imagen de un animal". Extraído de [11]

Figura 2: "Anotación manual aplicada a la imagen de un interior". Extraído de [11]

En la Figura 1 y 2 se muestran dos ejemplos de anotación manual, en las que se busca construir una colección de imágenes con etiquetas para ser usado con objeto de clasificación e investigación en su reconocimiento. Estos datos serán útiles para el aprendizaje supervisado y su evaluación.

Atendiendo a la granularidad en el procesado de las imágenes, se podrían distinguir además tres métodos principales para la anotación de imágenes:

- Clasificación: en este método se pretende dar una descripción de la imagen al completo. Se realiza mediante transformaciones globales que puedan decidir que tipo de imagen es la que se está visualizando teniendo en cuenta toda la escena.
- Detección: esta técnica consiste en recorrer toda la imagen aplicando múltiples ventanas detectoras de diferentes tamaños. Para cada ventana se aplica algún modelo comparativo que nos indica si existe el objeto que buscamos. El desplazamiento de la ventana tiene gran importancia, si la ventana es pequeña el proceso es más largo, pero si la ventana es grande corremos el riesgo de no detectar el objeto. En la Figura 3 se muestran ejemplos de detección de objetos y se su etiquetado.

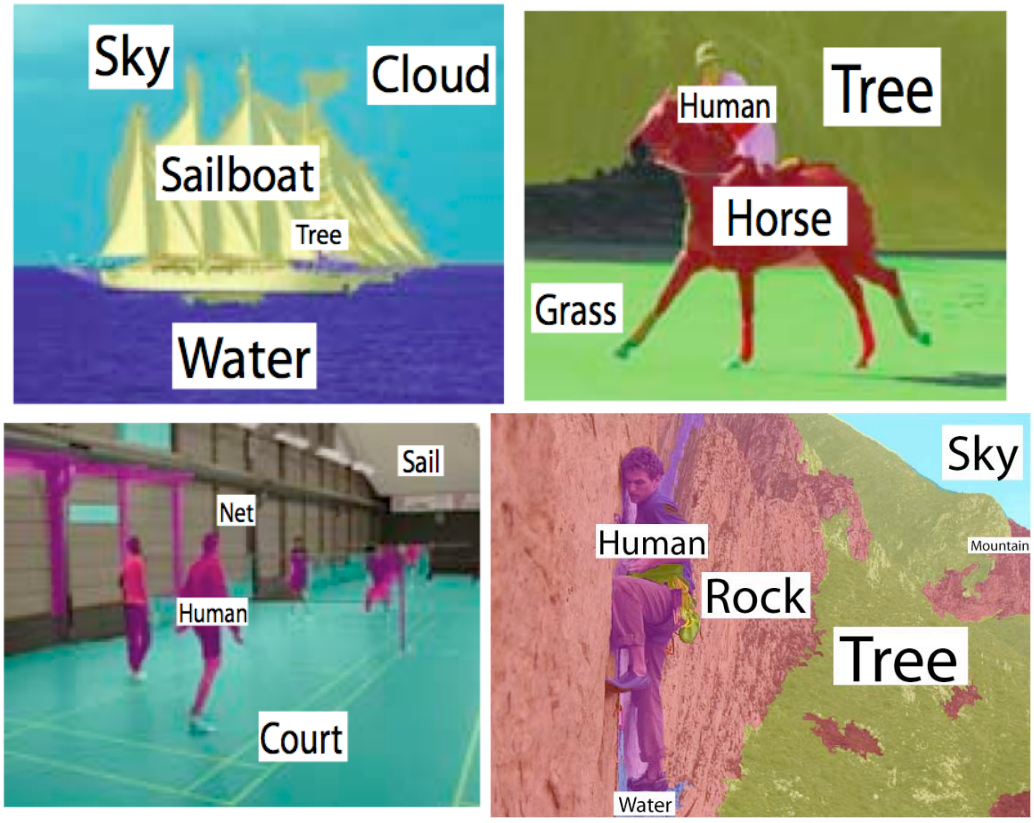


Figura 3. "Detección de imágenes con etiquetado". Extraído de [37]

- Segmentación basada en categorías: el objetivo es particionar una imagen en varias regiones, proporcionando a cada una de ellas una etiqueta de valor semántico. Depende de las características visuales de la imagen y de los resultados precisos de la segmentación. En la Figura 4 cada región segmentada corresponde a un objeto.

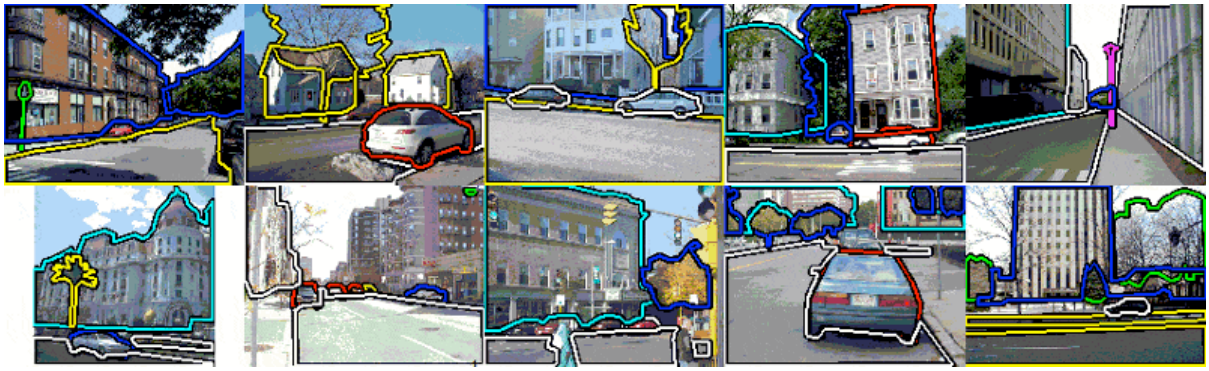


Figura 4. "Segmentación de imágenes". Extraído de [11]

2.3 Clasificación de imágenes

Las imágenes son clasificadas en clases usando unas características visuales de bajo nivel, las cuales se utilizan para discriminar correctamente la correspondencia de las imágenes a las clases. Esta clasificación es necesaria para dar a las imágenes una estructura en clases, la cual viene dada por las necesidades del usuario. Para ello, debemos hacer frente a la diversidad de clases posibles para hacer una correcta clasificación de imágenes, es decir, debemos escoger las clases que nos ofrezcan una mejor representación de la base de imágenes sobre la que queramos trabajar. Dicho conjunto de clases debe permitir a su vez, tener una clara dispersión estadística entre las características de la imagen, para que la clasificación sea lo más óptima posible.

2.3.1 Tipos de clasificadores de imágenes

Existen tres estrategias para la clasificación:

- La clasificación supervisada, también es conocida como clasificación con aprendizaje, en la que imponemos las especificaciones sobre los datos. Se tiene un conjunto de imágenes etiquetadas de entrenamiento en los que la característica puede o no aparecer, y se emplea para entrenar un clasificador que permita decidir la presencia de la característica. El clasificador después se aplica a todas las imágenes de la base de datos las cuales han sido anotadas con respecto a la presencia o ausencia de la característica (para su evaluación).

- La clasificación parcialmente supervisada, también es conocida como aprendizaje parcial, en los que existe una muestra de los objetos en alguna de las clases que han sido previamente definidas y un conjunto restante de contenidos no anotados.

- La clasificación no supervisada, también es conocida como clasificación sin aprendizaje, en la que los datos imponen las limitaciones a nuestra interpretación, donde simplemente se especifica el número de categorías deseadas. Su característica principal es que no existen etiquetas. Se intenta descubrir señales visuales que aparezcan consistentemente en las imágenes de la base de datos, y que permitan utilizar algoritmos de clasificación automática en los que los individuos más próximos se van organizando formando diferentes clases o categorías. Tarea muy relacionada con los algoritmos de agrupamiento.

2.3.2 Sistemas de reconocimiento de patrones

En el método de la clasificación digital de imágenes, se identifica el patrón visual asociado a cada categoría de acuerdo a una serie de parámetros como son el tono, la textura, la forma, el color, el contexto, la iluminación, la disposición, el tamaño... Después se identifican sobre las fotografías las superficies correspondientes a estas categorías, mediante su semejanza con el patrón-tipo previamente identificado. Por último es necesaria una verificación de los resultados. En contraposición al análisis digital de las imágenes, un humano no interpreta la imagen píxel a píxel, sino que observa el conjunto de información que muestran varias características básicas de la imagen y de los objetos que conforman la imagen, y a través de sus conocimientos previos, generaliza la clasificación de la imagen [12] [13].

El reconocimiento de patrones, también llamado lectura de patrones, identificación de figuras y reconocimiento de formas, consiste en la identificación de modelos de señal. Los patrones se obtienen a partir de los procesos de segmentación, y extracción de características donde cada objeto queda representado por una colección de descriptores. El sistema de reconocimiento, mostrado en la Figura 5, debe asignar a cada objeto su categoría o clase (conjunto de entidades que comparten alguna característica que las diferencia del resto).

Para poder reconocer los patrones se siguen los siguientes procesos:

- Adquisición de datos.
- Extracción de características.
- Toma de decisiones.

El punto esencial del reconocimiento de patrones es la clasificación: se quiere clasificar una señal dependiendo de sus características. Señales, características y clases pueden ser de cualquier forma.

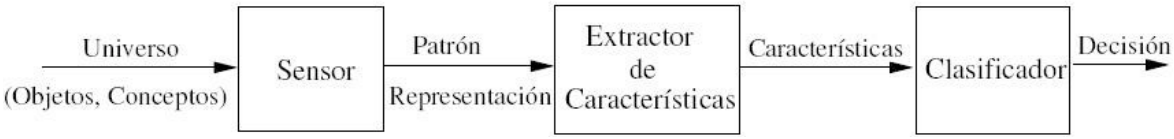


Figura 5. "Modelo de un reconocedor de patrones". Extraída de [14]

Una forma más detallada de un sistema de clasificación de patrones la realiza Shigeo Abe [15] en el cual se describen las siguientes fases:

- Determinación de características: se han de determinar las características adecuadas para la resolución del problema. Un número bajo de características podría llegar a ser crítico, ya que no podremos discriminar bien nuestras clases. Por tanto, se suele comenzar con un número elevado de características y después se eliminan las redundantes o las poco discriminativas.

- Normalización de los datos: en este proceso las muestras se transforman en un conjunto con un rango dinámico más pequeño. El objetivo fundamental de la normalización es evitar efectos indeseados ante la aparición de *outliers* o por los diferentes rangos dinámicos de las dimensiones de una característica.

- Optimización de características: se debe reducir el número de características al número óptimo para que estas puedan caracterizar bien nuestras clases. Si los resultados son malos, se debería realizar una nueva asignación de características analizando si las muestras de cada clase están correctamente agrupadas para facilitar su clasificación.

- División de datos: las muestras se deben dividir en dos subconjuntos los cuales deberían ser lo más parecidos estadísticamente posibles para tener una buena caracterización en ambos casos. Habitualmente, se utiliza del 60% al 80% de las muestras globales para un subconjunto con el cual haremos las tareas de entrenamiento, y el resto, del 40% al 20%, para el conjunto de *test*. Existe un tercer conjunto de datos llamado de validación, este se puede extraer del subconjunto de entrenamiento.

- Ajuste de parámetros libres: para seleccionar la configuración adecuada a nuestro sistema de clasificación necesitamos el conjunto de validación, después se entrenará la configuración óptima con el conjunto de entrenamiento.

- Evaluación del clasificador: finalmente, cuando se han ajustado los parámetros y realizado el entrenamiento, se procede a evaluar el clasificador con el conjunto de *test*.

2.4 Métodos de extracción de características

Estos métodos son el proceso de generar características que puedan ser usadas en la tarea de clasificación de los datos. En ocasiones viene precedido por un pre procesado de la señal, necesario para corregir posibles deficiencias en los datos debido a errores del sensor, o bien para preparar los datos de cara a posteriores procesos en las etapas de extracción de características o clasificación. Las características elementales están explícitamente presentes en los datos adquiridos, o bien pueden ser obtenidos con operaciones sencillas, por lo que pueden ser pasados directamente a la etapa de clasificación. Las características de medio nivel son derivadas de las elementales y son generadas por manipulaciones o transformaciones en los datos [14].

2.4.1 Características de bajo nivel

Una descripción de bajo nivel es la que describe los componentes individuales de una imagen. Se centran en cómo afectan y conforman la imagen, que a la imagen global. En contraposición, una descripción de alto nivel es la que describe la imagen en una componente general, detallando sus características y se centra en el sistema completo y sus objetivos. Estas características describen los rasgos principales de los objetos que componen la imagen, como puede ser el color, la textura, la forma, sus bordes, sus esquinas, etc.

Las características de bajo nivel se pueden extraer con diversa granularidad en una imagen. En particular, se pueden distinguir varios niveles [6]:

- Análisis orientado a la escena global. Aplicación de anotaciones de escenas básicas obtenidas mediante filtros o transformaciones globales.

- Segmentación de la imagen en varias regiones. Se aplican algoritmos de segmentación de imágenes, para extraer las características locales sobre dichas regiones (forma de la región, color, textura).

- Aplicación de rejillas para obtener características locales. Se divide la imagen en rejillas de tamaño fijo (ver Figura 6), para luego concatenar las características extraídas en cada una de las rejillas. Esta aproximación introduce cierta discriminación espacial.

Figura 6. "Imagen dividida en rejillas de tamaño fijo". Extraído de [39]

- Características sobre parches locales. Se localizan puntos clave (*keypoints*) en la imagen que luego serán descritos (ver Figura 6). Esta aproximación se describirá en detalle en la siguiente sección.

Figura 7. "Detección de puntos clave". Extraido de [29]

2.4.2 *SIFT*

SIFT (*Scale Invariant Feature Transform*) [29] es un algoritmo que permite detectar y describir las características de una imagen en regiones locales de una forma invariantes a la escala. *SIFT* detecta y utiliza un número grande de características de las imágenes, lo que reduce la contribución de los errores causados por estas variaciones locales en el error promedio de todos los errores de características coincidentes.

A continuación presentamos el algoritmo extendido para generar un conjunto de características sobre una imagen. El algoritmo tiene dos pasos principales:

- Detección de regiones locales de interés: fase en la que se detectan regiones locales de la imagen consideradas "de interés" para su clasificación (*salient regions*). El proceso de detección puede, a su vez, dividirse en varios pasos:

    - Detección de extremos en el espacio de escala: el algoritmo busca regiones de interés sobre las localizaciones de las imágenes a diferentes escalas. Es implementado eficientemente usando una función de diferencia de Gaussianas para identificar puntos de interés potenciales que son invariantes a la escala y a la orientación. La imagen inicial es repetidamente convolucionada con Gaussianas para producir el conjunto de imágenes espacio escala. Las imágenes Gaussianas adyacentes son substraídas para producir las imágenes Diferencia de Gaussiana (*DoG*). Después de cada octava, la imagen Gaussiana es submuestreada por un factor de 2, y el proceso es repetido (ver Figura 8). El objetivo de la detección es encontrar extremos en el espacio *DoG*, que se corresponderán con puntos de interés (*keypoints*).

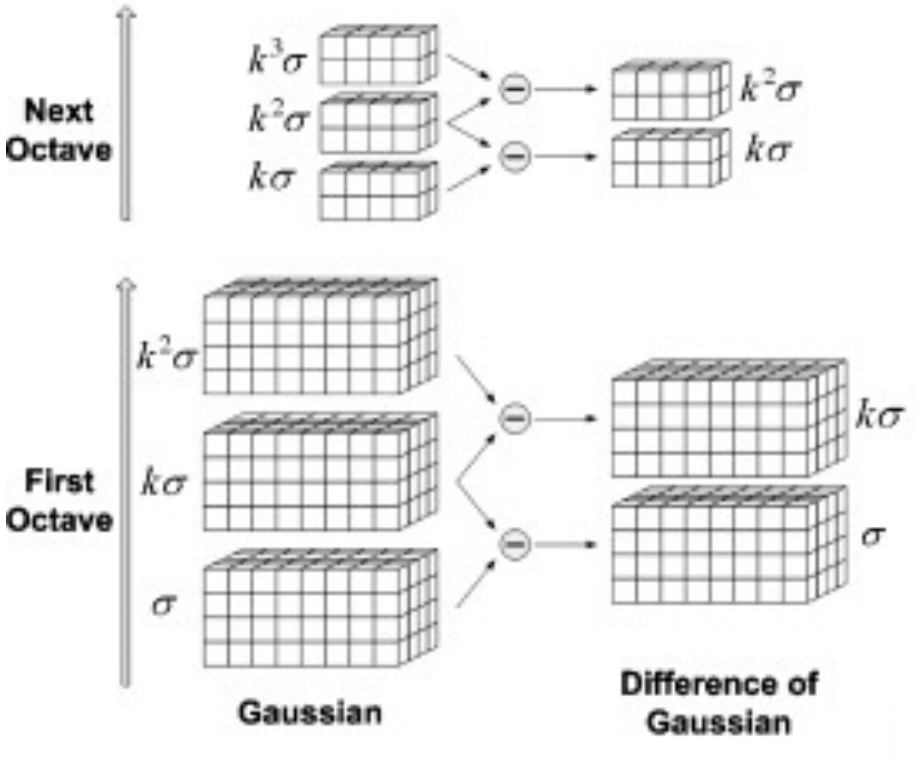


Figura 8. "Representación del proceso de Diferencia de Gaussianas". Extraído de [38]

- Localización de puntos clave: para cada localización detectada, se crea un modelo detallado que incluye la localización y la escala. Los puntos clave son seleccionados basados en medidas de su estabilidad.

- Asignación de la orientación: a cada localización de puntos clave se le asigna una o más orientaciones basadas en direcciones de gradientes de imagen local. Todas las operaciones futuras son realizadas en los datos de la imagen que han sido transformados en relación a la orientación asignada, escala y localización para cada característica, proporcionando así invarianza a dichas transformaciones. En la Figura 9 se muestra una imagen a la que se le ha localizado los puntos clave y se le han asignado una orientación.

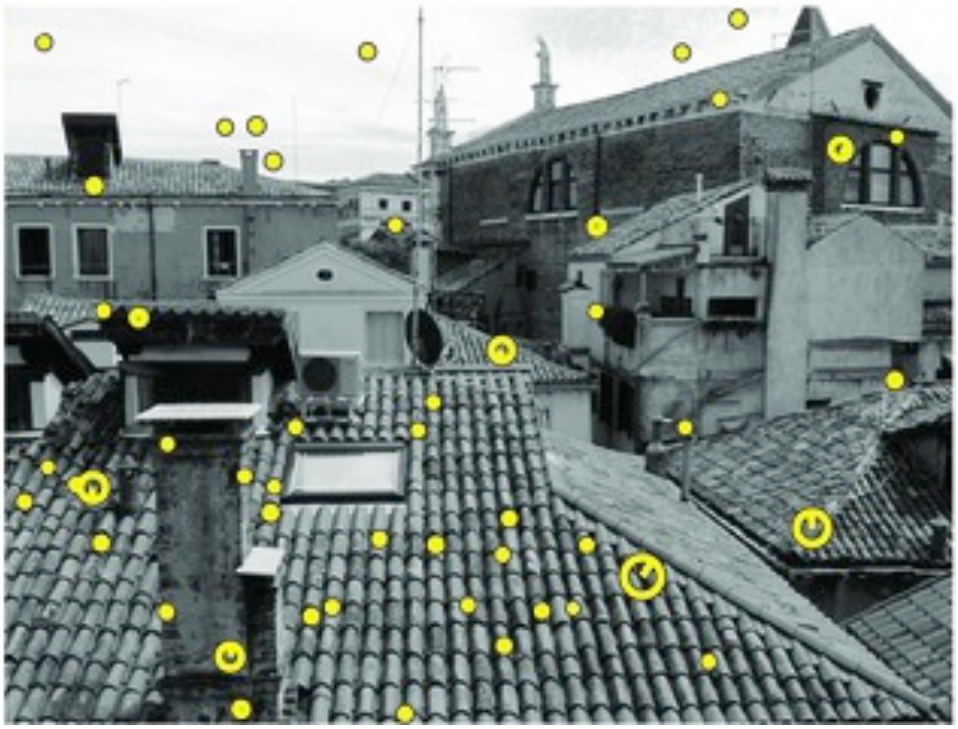


Figura 9. "Localización de puntos clave y asignación de la orientación". Extraído de [7]

- Descripción de las regiones locales de interés: los gradientes locales de imagen son medidos a la escala seleccionada en la región alrededor de cada punto clave. Estos son transformados en una representación que tiene en cuenta significativos niveles de distorsión de forma local, así como cambios en la iluminación. Inicialmente, se calcula la magnitud del gradiente y la orientación de cada punto en una región alrededor de la localización del punto clave. A estos se les asigna un peso con una ventana Gaussiana, indicado por un círculo superpuesto (según el peso asignado, el círculo tendrá un radio mayor o menor). Estas muestras son después acumuladas en histogramas de orientaciones, resumiendo los contenidos sobre subregiones resultantes de una división de la región en una celda 4x4. La longitud de cada flecha corresponde a la suma de las magnitudes de gradiente en dicha dirección dentro de la región. La Figura 8 muestra, una técnica usada en *SIFT* para reconocimiento de regiones llamada Diferencia de Gaussianas (*DoG*), la cual opera sobre 8 orientaciones del plano en una celda espacial local de 4x4, dando como salida un vector de 128 dimensiones (16 celdas para las 8 orientaciones) [28].

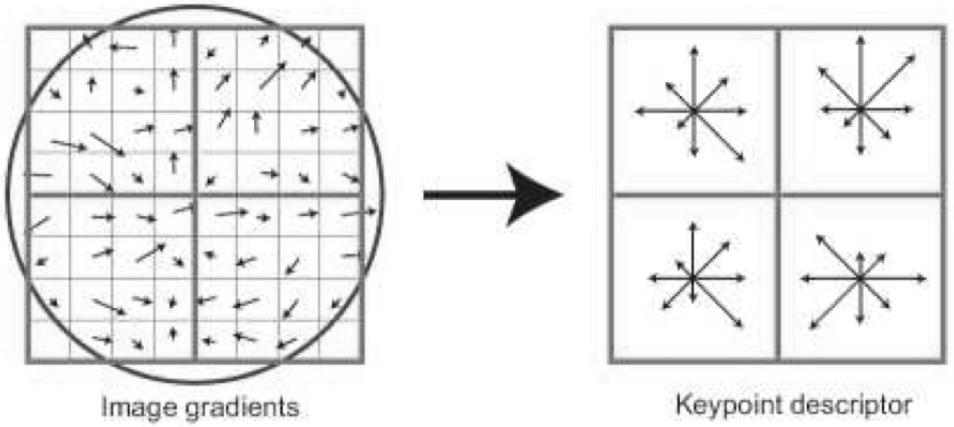


Figura 10. "Descriptor SIFT". Extraído de [3]

2.4.3 *Descriptor Gist*

A. Introducción

El *Gist descriptor* es un modelo de reconocimiento de escenas del mundo real, propuesto en [2], que no pasa por la segmentación ni por el procesado de objetos. En ese sentido, se estima la estructura o "forma de una escena" con unas dimensiones perceptivas específicamente dedicadas a describir las propiedades espaciales de la escena. Se demuestra que las propiedades que describen la escena, denominadas atributos de *Spatial Envelope*, puede ser estimadas de manera fiable utilizando la información espectral y localizada en porciones grandes de la imagen [2].

B. Estudios sobre el reconocimiento de escenas

Este tipo de descriptores debe su nombre, *Gist*, a que tratan de buscar la esencia de la imagen, es decir, el reconocimiento del tipo de escena que se está visualizando. En anteriores estudios de [2], se propone que el ser humano es capaz de reconocer la esencia de una imagen tan rápidamente como la de un simple objeto [25], y que, de hecho, la rapidez del reconocimiento de escenas puede ser debida a la disposición espacial de los objetos y al plano de la escena [27]. En resumen, cuando se ve una escena por un tiempo corto, extraemos visualmente suficiente información para reconocer con exactitud sus propiedades funcionales y categóricas, mientras pasamos por alto la mayor parte de la información perceptual sobre los objetos y sus ubicaciones. Por tanto, la representación semántica primaria parece basarse en una configuración de baja resolución espacial.

C. La forma de la escena. Definición de *Spatial Envelope*

De todas las propiedades visuales, la forma o silueta es la propiedad que ayuda a identificar un estímulo visual. Si bien una escena es un cúmulo de objetos que, en consecuencia, se define según la disposición y variedad de ellos; sin embargo, el *Gist descriptor* considera la escena como un objeto individual con una única forma.

El *Spatial Envelope* es una descripción de la representación de la escena a partir de propiedades perceptivas que puede ser reconocidas a partir de cálculos sencillos, y que pueden describir el espacio de la escena. Recibe este nombre ya que, principalmente, se refiere a las cualidades del espacio. El *Spatial Envelope* está representado por la relación entre los contornos de las superficies y sus atributos, incluyendo el patrón generado por las texturas [2].

D. Atributos a modelar con el *Spatial Envelope*

Los autores de *Gist descriptor* quisieron modelar las imágenes a través de una serie de atributos de medio nivel. La respuesta o medida de dichos atributos sobre una imagen constituirán la características sobre las que ellos realizan su clasificación en una categoría.

- Grado de Naturaleza (*Naturalness*):
  La estructura de una escena es muy diferente entre entornos artificiales y naturales. Las líneas rectas horizontales y verticales dominan las estructuras artificiales, mientras que la mayoría de los paisajes naturales tienen zonas de texturas y contornos con forma ondulada. Por lo tanto, las escenas con bordes sesgados hacia orientaciones vertical y horizontal tendrán un bajo grado de naturaleza (Figura 11.a), mientras que en las escenas con alto grado de ondas tendrán un alto grado de naturaleza (Figura 11.b).

a)  b) 

Figura 11. "Ejemplos de grado de naturaleza a) Bajo grado b) Alto grado". Extraídas de [9]

- Grado de Apertura (*Openness*):

  La existencia de la línea del horizonte y la falta de objetos en los límites de la imagen confieren a la escena de un alto grado de apertura (Figura 12.b). El grado de apertura de una escena disminuye cuando el número de elementos en los bordes aumenta (Figura 12.a).

a)  b) 

Figura 12. "Ejemplos de grado de apertura a) Bajo grado b) Alto grado". Extraídas de [9]

- Grado de Aspereza (*Roughness*):

  Depende del tamaño de los elementos en cada escala del espacio, su capacidad de construir elementos complejos y sus relaciones entre elementos que también pertenecen a otras estructuras. La aspereza tiene estrecha relación con la dimensión fractal de la escena y, por tanto, de su complejidad. La comparativa gráfica está en Figura 13.

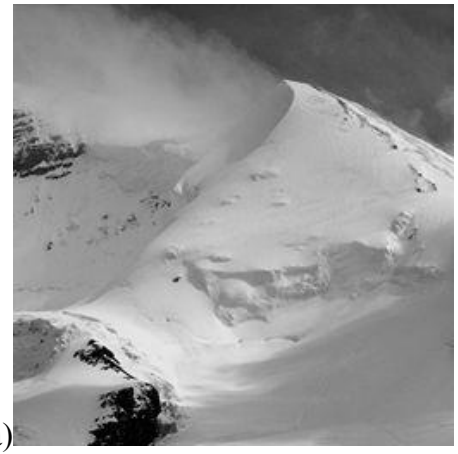
a)  b) 

Figura 13. "Ejemplos de grado de aspereza a) Bajo grado b) Alto grado". Extraídas de [9]

- Grado de Expansión (*Expansion*):

  La convergencia de las líneas paralelas da la percepción de profundidad al espacio. Una visión plana de un edificio tiene un bajo grado de expansión (Figura 14.a). Por el contrario, una calle con largas líneas de fuga tendrá un alto grado de expansión (Figura 14.b).

a)  b) 

Figura 14. "Ejemplos de grado de expansión a) Bajo grado b) Alto grado". Extraídas de [9]

- Grado de Rugosidad (*Ruggedness*):

  La rugosidad se refiere a la desviación de la tierra con respecto al horizonte. Entornos con rugosidad producen contornos oblicuos en la imagen y ocultan la línea del horizonte. La mayoría de los entornos artificiales se construyen sobre una superficie plana (Figura 15.a). Por lo tanto, entornos con robustez son en su mayoría natural (Figura 15.b).

a)  b) 

Figura 15. "Ejemplos de grado de rugosidad a) Bajo grado b) Alto grado". Extraídas de [9]

E. Filtrado de Gabor

Gist Descriptor utiliza para sus salidas la creación de un filtro de tipo Gabor. En el procesamiento de la imagen, es un filtro lineal utilizado para la detección de bordes. Las representaciones de las frecuencias y las orientaciones de los filtros de Gabor son similares a las del sistema visual humano, y se ha encontrado que es particularmente apropiado para la representación y discriminación de texturas. En el dominio espacial, un filtro de Gabor de dos dimensiones es una función *kernel* gaussiana modulada por una onda plana sinusoidal. La transformada de Fourier de un filtro de Gabor son gaussianas centradas en la frecuencia de la función sinusoidal (siendo estas gaussianas la transformada de Fourier de la gaussiana temporal o espacial). La Figura 16 muestra un ejemplo de filtro de Gabor en un gráfico tridimensional.

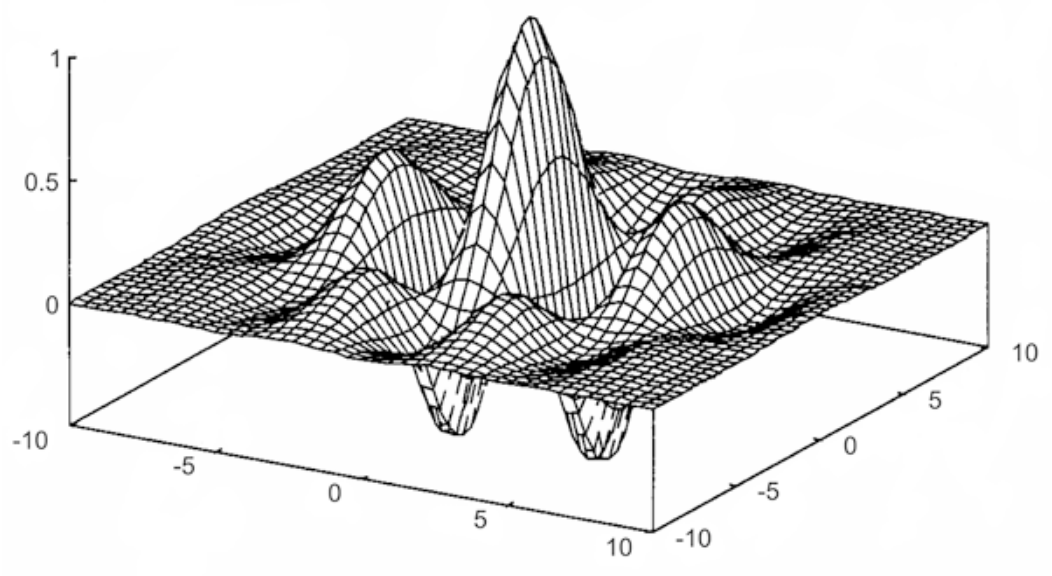


Figura 16. "Filtro de Gabor". Extraído de [40]

Se puede llegar a este resultado empleando la propiedad de convolución de la Transformada de Fourier, que transforma los productos en convoluciones. Así, la transformada de la respuesta de impulso de Gabor es la convolución de la transformada de la función sinusoidal y de la transformada de la función gaussiana [31].

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

$$x' = x\cos\theta + y\sin\theta \qquad y' = -x\sin\theta + y\cos\theta$$

Ecuación 1. "Función compleja que conforma el filtro de Gabor"

Haciendo pruebas, se puede comprobar como los filtros de Gabor tienen una notable similaridad con los atributos de *Spatial Envelope* que antes se han comentado.

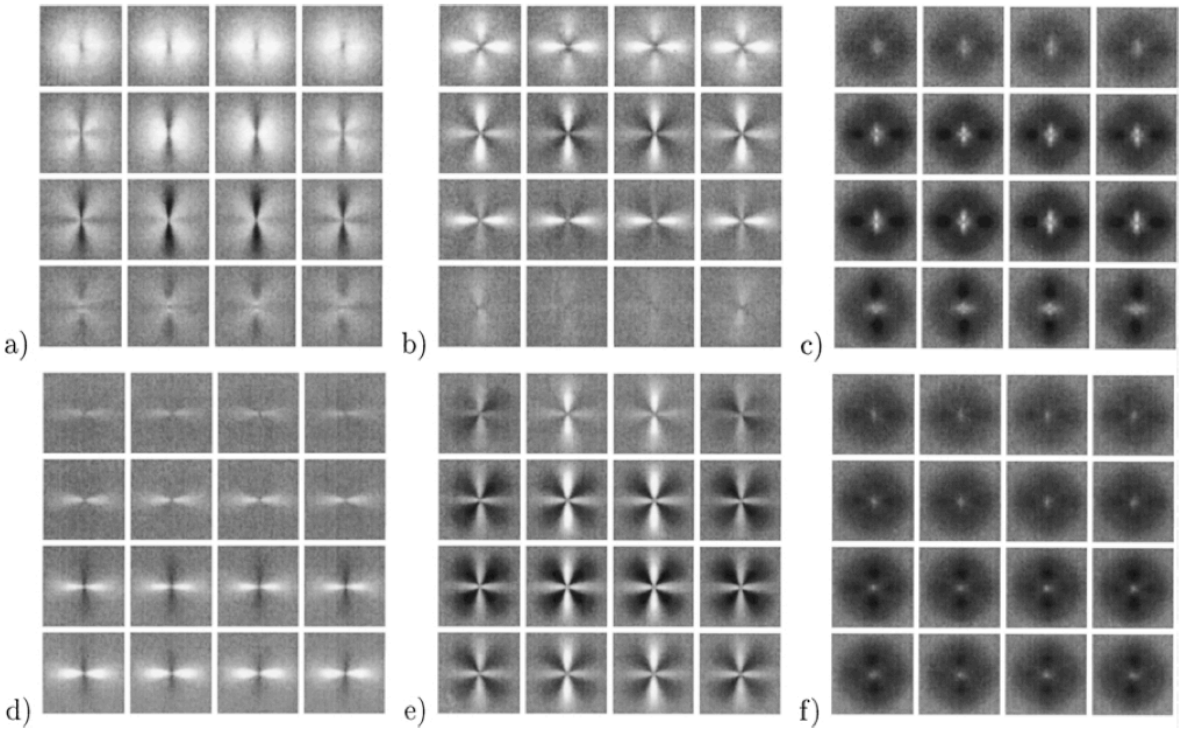


Figura 17. "Ejemplos de Filtros de Gabor aplicadas a imágenes". Extraído de [2]

En la Figura 17 se pueden observar ejemplos de Filtros de Gabor aplicadas a imágenes de la base de datos. Las imágenes corresponden a escenas de la naturaleza con alto grado de: a) apertura , b) robustez , c) rugosidad. Y la segunda fila de imágenes corresponde a escenas de escenarios no naturales, realizados por el hombre, que tienen alto grado de , d) apertura, e) expansión y f) rugosidad.

F. Representaciones basadas en imagen

Siendo $i(x,y)$* la distribución de intensidad de la imagen en las variables de espacio (x,y), f_x y f_y las variables de las frecuencias espaciales, $h(x,y)$ una ventana circular de tipo Hann que reduce el efecto de los bordes, con un tamaño igual a las dimensiones de la imagen completa, y $h_r(x,y)$ una ventana de tipo Hamming con un soporte circular de radio r, mucho más pequeña que las dimensiones de la imagen, definimos:

* a $i(x,y)$ se le aplica un pre filtrado para reducir los efectos de iluminación y prevenir algunas zonas de la imagen donde domine la energía. El pre filtrado consiste en una normalización local de la varianza de la intensidad

$$i'(x,y) = \frac{i(x,y) * h(x,y)}{\epsilon + \sqrt{[i(x,y) * h(x,y)]^2 * g(x,y)}}$$

Ecuación 2. "Pre filtrado de la intensidad de la imagen"

- la Transformada Discreta de Fourier (DFT) de una imagen como:

$$I(f_x, f_y) = \sum_{x,y=0}^{N-1} i(x, y)h(x, y)e^{-j\,2\pi(f_x x + f_y y)}$$

Ecuación 3. "DFT para una imagen"

- y la Transformada Enventanada de Fourier (WFT) de una imagen como:

$$I(x, y, f_x, f_y) = \sum_{x',y'=0}^{N-1} i(x', y')\, h_r(x' - x, y' - y)e^{-j\,2\pi(f_x x' + f_y y')}$$

Ecuación 4. "WFT para una imagen"

En ambas ecuaciones, podemos descomponer los términos en amplitud y fase de la Transformada de Fourier. A partir de esta división podemos definir los conceptos de energía espectral y de espectrograma:

- para DFT tenemos que la energía espectral total es:

$$A(f_x, f_y)^2 = |I(f_x, f_y)|^2$$

Ecuación 5. "Energía espectral"

- y para WFT tenemos que el espectrograma es:

$$A(x, y, f_x, f_y)^2 = |I(x, y, f_x, f_y)|^2$$

Ecuación 6. "Espectrograma"

En su implementación, los autores consideraron ψ_i como un conjunto de filtros de Gabor para diferentes escalas y orientaciones.

Usando una técnica habitual en el reconocimiento de patrones, como es la Karhunen-Loeve Transform (*KLT*) de Karhunen et Loeve [41], podemos extraer características y reducir las dimensiones de estos descriptores. Entonces, tomando ψ_i como las bases ortogonales de la *KLT* y *N* como el número de funciones usadas para aproximar y determinar las dimensiones de cada representación, podremos descomponer las anteriores ecuaciones:

- la energía espectral como:

$$A(f_x, f_y)^2 \simeq \sum_{i=1}^{N} v_i \, \psi_i(f_x, f_y)$$

Ecuación 7. "Descomposición de la energía espectral"

donde v_i proporciona información estructural sin traducir y contiene una descripción de baja resolución del espectro de energía de la imagen.

- y el espectrograma como:

$$A(x, y, f_x, f_y)^2 \simeq \sum_{i=1}^{N} w_i \, \Psi_i(x, y, f_x, f_y)$$

Ecuación 8. "Descomposición del espectrograma"

donde w_i proporciona información estructural con una descripción de la disposición espacial. Debido a la reducida dimensión de *N*, la representación tiene una resolución baja en ambos dominios espectrales y espaciales.

G. *Discriminant Spectral Template*

La estimación de los atributos del *Spatial Envelope* de la características basadas en la imagen puede ser resuelta usando diferentes técnicas de regresión. En el caso de un simple regresor lineal, la estimación de un atributo del *Spatial Envelope s* (naturaleza, apertura, aspereza, expansión y rugosidad):

- para las características del espectro global **v** de una escena se describe como:

$$\hat{s} = \mathbf{v}^T \mathbf{d} = \sum^{N} v_i \, d_i = \iint A(f_x, f_y)^2 \, DST(f_x, f_y) \, df_x \, df_y$$

Ecuación 9. "Estimación para la energía espectral"

La *DST* (*Discriminant Spectral Template*) es una función que describe cómo cada componente espectral contribuye a las atributos del *Spatial Envelope.*

$$DST(f_x, f_y) = \sum^{N} d_i \, \psi_i(f_x, f_y)$$

Ecuación 10. "Discriminant Spectral Template"

- para las características del espectrograma **w** de una escena se describe como:

$$\hat{s} = \mathbf{w}^T \mathbf{d} = \sum_{i=1}^{N} w_i \, d_i = \sum_x \sum_y \iint A(x, y, f_x, f_y)^2 \, WDST(x, y, f_x, f_y) \, df_x \, df_y$$

Ecuación 11. "Estimación para el espectrograma"

La *WDST* (*Windowed Discriminant Spectral Template*) es la función que describe cómo los componentes espectrales en los diferentes espacios contribuyen a las atributos del *Spatial Envelope*.

$$WDST(x, y, f_x, f_y) = \sum_{i=1}^{N} d_i \, \Psi_i(x, y, f_x, f_y)$$

Ecuación 12. "Windowed Discriminant Spectral Template"

H. Modelado de atributos

Para cada imagen estimamos el atributo del *Spatial Envelope* como $\hat{s}_t = \boldsymbol{v}_t^T \boldsymbol{d} + d_0$ para DST, o $\hat{s}_t = \boldsymbol{w}_t^T \boldsymbol{d} + d_0$ para WDST. Ambos tipos de atributos son el resultado de multiplicar dos vectores:

- $\boldsymbol{v}$ o $\boldsymbol{w}$: son coeficientes obtenidos que dependen únicamente de las características de una imagen particular.

- $\boldsymbol{d} = \{d_i\}$: es un vector asociado a cada atributo del *Spatial Envelope*, que difiere según el atributo que vayamos a representar, dando más peso a las bases ortogonales de la *KLT*.

$\boldsymbol{d}$ se determina durante la fase de entrenamiento anotando imágenes según los diferentes atributos del *Spatial Envelope*. Después se obtiene $\boldsymbol{d}$ para cada atributo mediante regresión. De esta forma se obtiene la pseudoinversa que minimiza el error cuadrático medio como $\boldsymbol{d}_1 = (\boldsymbol{V}_1 \boldsymbol{V}_1^T)^{-1} \boldsymbol{V}_1 \boldsymbol{s}$ para *DST*, o $\boldsymbol{d}_1 = (\boldsymbol{W}_1 \boldsymbol{W}_1^T)^{-1} \boldsymbol{W}_1 \boldsymbol{s}$ para *WDST*, el vector $\boldsymbol{d}_1$ contiene los parámetros de *DST* o de *WDST* y la constante d_0 que no afecta a la organización y discriminación de los atributos [28]. En este proyecto se han empleado los vectores $\boldsymbol{d}$ propuestos por los autores de [2].

I. *Gist* para clasificación de imágenes

A diferencia del trabajo original, en nuestra aplicación no es necesario calcular o estimar dichos atributos de medio nivel. En nuestro caso, necesitamos un vector que modele el *Spectral Envelope* de una imagen y que puede utilizarse mediante técnicas de aprendizaje máquina para clasificarla en una categoría. Es por ello que, en el presente proyecto se utilizarán las variables $\boldsymbol{v}$ y $\boldsymbol{w}$ como vector de características. Dado un conjunto de N filtros de Gabor (con varias orientaciones y escalas), cada componente del vector será:

- para *Discriminant Spectral Template*:

$$v_i = \int\int A(f_x, f_y)^2 \psi_i(f_x, f_y)\, df_x\, df_y$$

Ecuación 13. "Componentes del vector DST"

- para *Windowed Discriminant Spectral Template*:

$$w_i = \sum_x \sum_y \int\int A(x, y, f_x, f_y)^2 \psi_i(x, y, f_x, f_y)\, df_x\, df_y$$

Ecuación 14. "Componentes del vector WDST"

2.4.4 *Bag of words*

A. Introducción

El modelo de Bolsa de Palabras, *Bag of words* [42], es un método usado para la clasificación de imágenes, que intenta resolver la identificación de objetos en imágenes reales, generalizándolas a través de las variaciones del objeto. El método del *Bag of words* se basa en la cuantificación vectorial sobre parches locales de una imagen, lo que permite asociar a cada descriptor con una palabra visual, y estudiar finalmente cada imagen como una bolsa de dichas palabras. Las principales ventaja de utilizar este método son su facilidad de uso, su eficiencia computacional y su invarianza frente a diversas transformaciones derivadas del empleo habitual de características que ofrecen esa propiedad.

B. Implementación del modelo

Los pasos que se deben seguir para la implementación de este método son:

- Detección y descripción de los parches de la imagen.
- Asignación de los parches en un determinado grupo, llamado vocabulario, con un algoritmo de cuantificación vectorial.
- Construir el *Bag of words*, el cual cuenta el número de parches asignados a cada grupo.
- Aplicar un clasificador, que trate el *Bag of words* como un vector de características, y así determinar a qué clase se le asignará la imagen.

Estos pasos están diseñados para maximizar la precisión de la clasificación y reducir el coste computacional. Por lo tanto, los descriptores extraídos en el primer paso deben de ser invariantes a transformaciones geométricas (escalado y rotación), y a cambios de iluminación. El vocabulario utilizado en el segundo paso debe ser lo suficientemente grande para distinguir cambios en diferentes partes de la imagen, pero no tan grande

como para distinguir variaciones irrelevantes, tales como ruido [3]. En la Figura 18 se propone un ejemplo esquemático del modelo de Bolsa de Palabras.

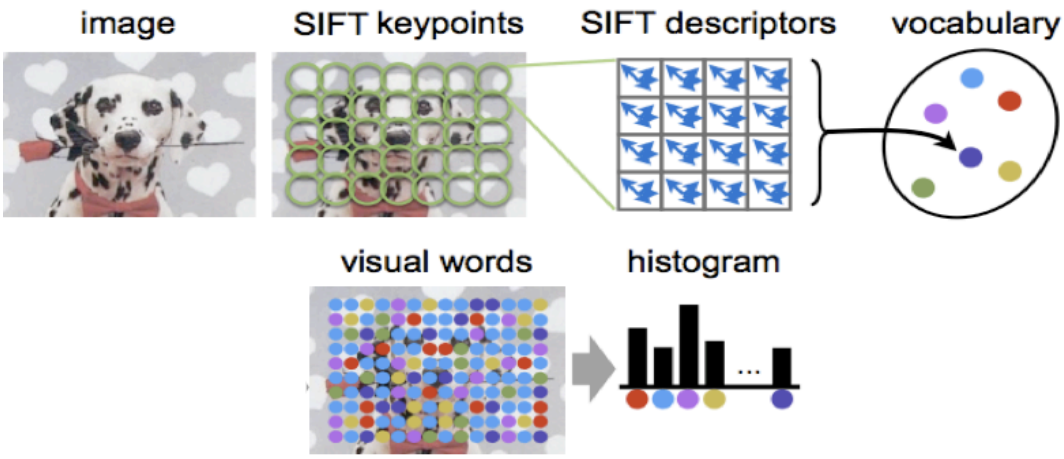


Figura 18. "Bag of words para una imagen con descriptores SIFT". Extraído de [20]

C. Generación del vocabulario

El vocabulario es la manera de construir nuestro vector de características para la clasificación, pues relaciona los descriptores de las imágenes de nuevas consultas a los descriptores previamente vistos en los entrenamientos. En la Figura 19 se muestra un ejemplo de extracción de descriptores de una imagen, más tarde, algunos de estos formarán parte del vocabulario.

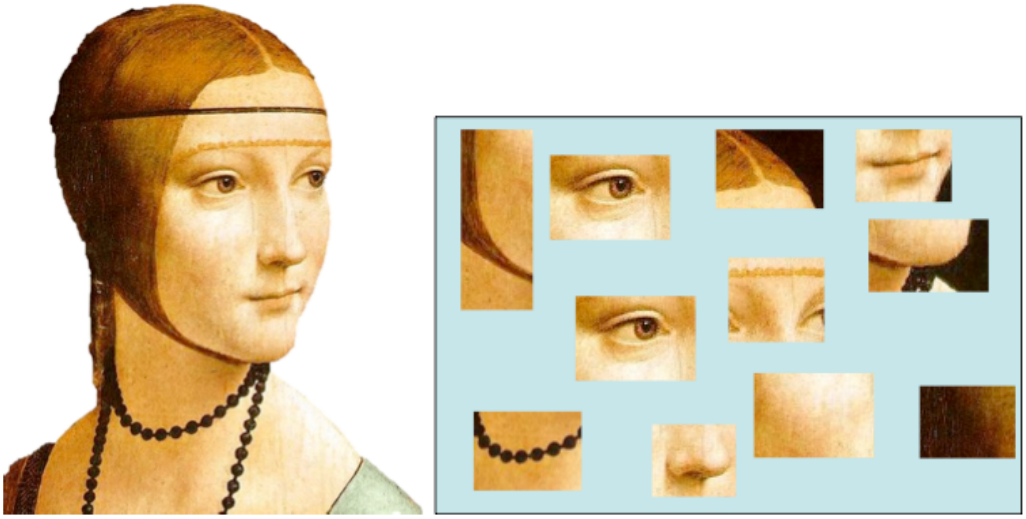


Figura 19. "Extracción de palabras clave de una imagen". Extraído de [43]

Una vez todas las regiones de interés de las imágenes han sido descritas, se genera un vocabulario representativo de las características que aparecen en las imágenes (en la Figura 20 se observa cómo a partir de los descriptores se procede a crear un vocabulario). Para ello, a través de métodos de agrupamiento se organizan los datos en ciertos grupos que se corresponden con palabras visuales de dicho vocabulario. Cada palabra visual, por lo tanto, constituirá la caracterización de un patrón visual. Posteriormente, cada descripción, asociada a un punto de interés en una imagen, se proyectará sobre el vocabulario asignándosele la palabra más parecida. La generación de vocabularios permite reducir la dimensionalidad de los datos de entrada al asignar cada descriptor a un único *codeword* (128 en el caso de descriptores *SIFT*) y, además, permite clasificar imágenes a través de un número variable de descriptores locales.

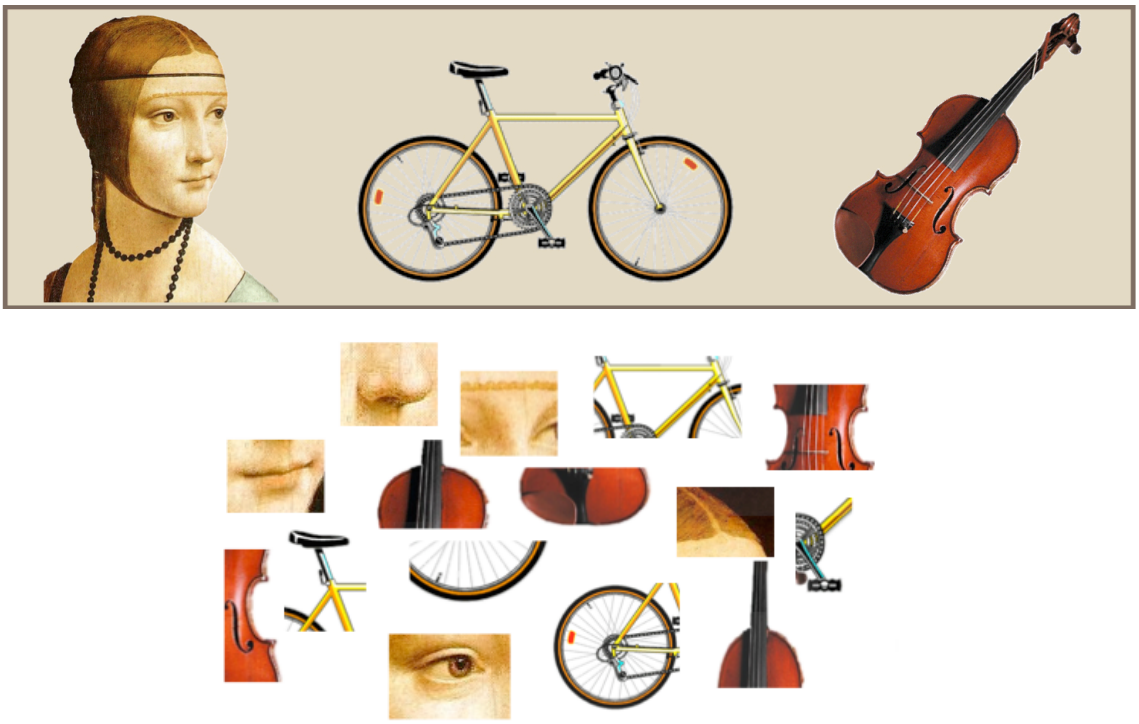


Figura 20. "Creación de un vocabulario a partir de imágenes". Extraído de [43]

D. Algoritmo *k-means*

En la práctica, para generar vocabularios se emplean algoritmos de agrupamiento como el algoritmo k-*means* [45], dada la gran cantidad de datos y la elevada dimensión de los mismos. El agrupamiento consiste en dividir un conjunto de datos sin etiquetar en subconjuntos según la distribución de los datos. El algoritmo *k-means* procesa los puntos asignándolos al centroide del grupo más cercano y recalcula de nuevo estos centroides en bucle, y en cada iteración van cambiando su posición, ajustándose al centro teórico del grupo. Se ha de definir el parámetro k, que es el número de centroides, que deberá ser igual al número de clases. El algoritmo *k-means* converge sólo a un óptimo local.

E. Histograma normalizado

Una vez que los descriptores asociados a cada región de interés de la imagen han sido asignados al vocabulario más cercano es necesario normalizar los datos para el óptimo uso de algoritmos de aprendizaje máquina para la clasificación de las imágenes. El primer paso para el normalizado es contar la aparición de todas las palabras del vocabulario en una imagen, y después dividir estas apariciones entre el número total de palabras encontradas. Así, con independencia del número de palabras presentes en cada imagen, todos los histogramas tendrán el mismo tamaño y cumplirán que la suma de los valores de todas sus barras será igual a 1 [6]. Vemos en la Figura 21 el proceso de normalización para un histograma.

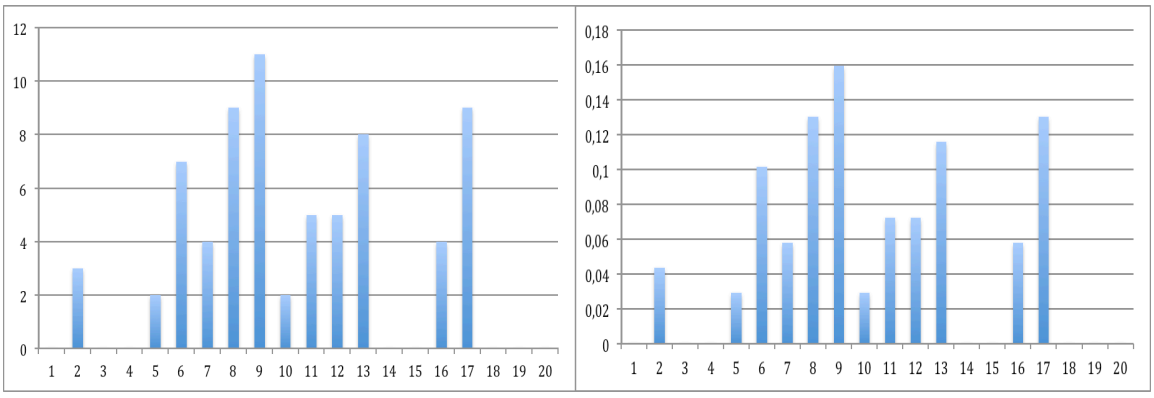


Figura 21. "Ejemplo de normalización de histograma"

2.5 Algoritmos de aprendizaje máquina

El aprendizaje automático consiste en el diseño y desarrollo de algoritmos que toman los datos y aprenden los patrones subyacentes a partir de dichas características. Estos mecanismos pretenden generalizar los comportamientos de los diferentes datos de entrada, que, en un principio, muestran una estructura desorganizada. Un aspecto importante de la investigación en el aprendizaje máquina es el diseño de algoritmos que reconozcan patrones complejos y sean capaces de tomar decisiones inteligentes sobre la base de los datos de entrada. Una dificultad fundamental es que el conjunto de todos los posibles comportamientos, dadas todas las posibles entradas, es demasiado grande para ser incluido en el conjunto de entrenamiento. Por lo tanto, el clasificador debe generalizar, a partir de los datos de entrenamiento, una salida útil en los casos nuevos.

Algunos sistemas de aprendizaje de máquina intentan eliminar la necesidad de la intuición humana en el análisis de datos, mientras que otros adoptan una colaboración entre el humano y la máquina. La intuición humana no puede, sin embargo, ser completamente eliminada, puesto que el diseñador del sistema debe especificar cómo los datos van a ser representados y qué mecanismos se han de utilizar para buscar una caracterización de los datos [26].

2.5.1 Support Vector Machine (SVM)

- *SVM* Lineal:

SVM (Máquina de Vector Soporte) [22] es una técnica útil para la clasificación de datos. Dado un conjunto de entrenamiento podemos etiquetar las clases y entrenar una *SVM* para construir un modelo que prediga la clase de una nueva muestra. Una *SVM* es un modelo que representa los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible mediante una frontera de decisión. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad a la frontera de decisión, pueden ser clasificadas a una u otra clase. Una buena separación entre las clases permitirá una clasificación correcta. Este tipo de algoritmos buscan el

hiperplano que tenga la máxima distancia con los puntos que estén más cerca de él mismo, si esto se cumple se denomina hiperplano óptimo. De esta forma, los puntos del vector etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado. A los puntos más cercanos al hiperplano se les conoce como vectores soporte [24] (ver Figura 22).

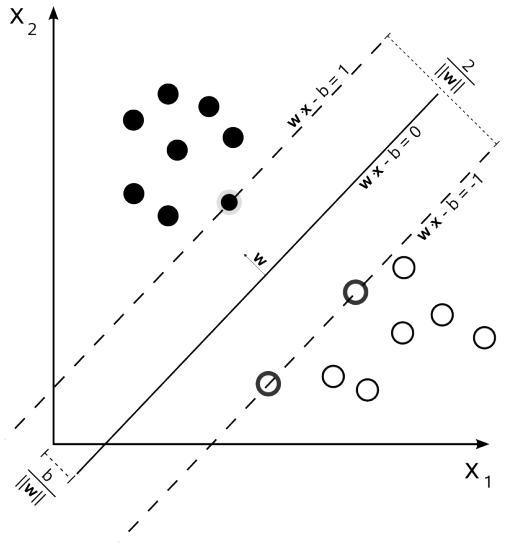


Figura 22. "Ejemplo de los márgenes de SVM". Extraído de [24]

Dado un conjunto de etiquetas de entrenamiento (x_i, y_i), $i = 1, 2, \ldots, l$ donde x_i es un vector de datos de dimensión p ($x_i \in R^n$) y y_i indica a la clase que pertenece el dato x_i denotándolo como *0* o *1* ($y_i \in \{0, 1\}^l$), se desea encontrar un hiperplano óptimo que minimice la probabilidad de errores del clasificador, para ello se han de encontrar los valores óptimos al vector de pesos w y del umbral b de la ecuación $w^T x - b = 0$, que sea:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

Ecuación 15. "Hiperplano óptimo SVM"

para $i = 1, 2, \ldots, l$ y $\varepsilon_i \geq 0$ para todo i, y tal que el vector de pesos w y la anchura de la función o holgura ε_i, minimicen la función de coste:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

Ecuación 16. "Función de coste SVM"

C > 0 es el parámetro de penalización de los errores de clasificación, es seleccionado por el usuario. Este parámetro controla el compromiso entre la complejidad de la máquina y el número de puntos no separables [22].

- *SVM* no lineal: el truco del *kernel*

Por otra parte, cuando nos encontramos con un problema no lineal, en el que las muestras no son linealmente ni poligonalmente separables, se necesita una herramienta que transforme el espacio de muestras en uno de orden superior para poder obtener una solución lineal a nuestro problema no lineal. Para resolver este problema se utiliza el truco del *kernel*. Las funciones que utilizan el truco del *kernel* pueden ser expresadas a partir de un producto entre dos vectores, como $K(x_i, x_j) \equiv \varphi(x_i)\ T\varphi(x_j)$. Estas funciones transforman el espacio de muestras en uno de orden superior, de este modo un algorimo no lineal puede ser transformado en lineal en el espacio φ (ver Figura 23). Sin embargo, como se usan los *kernels*, la función φ nunca se calcula explícitamente, porque el espacio de altas dimensiones puede ser infinitamente dimensional [26].

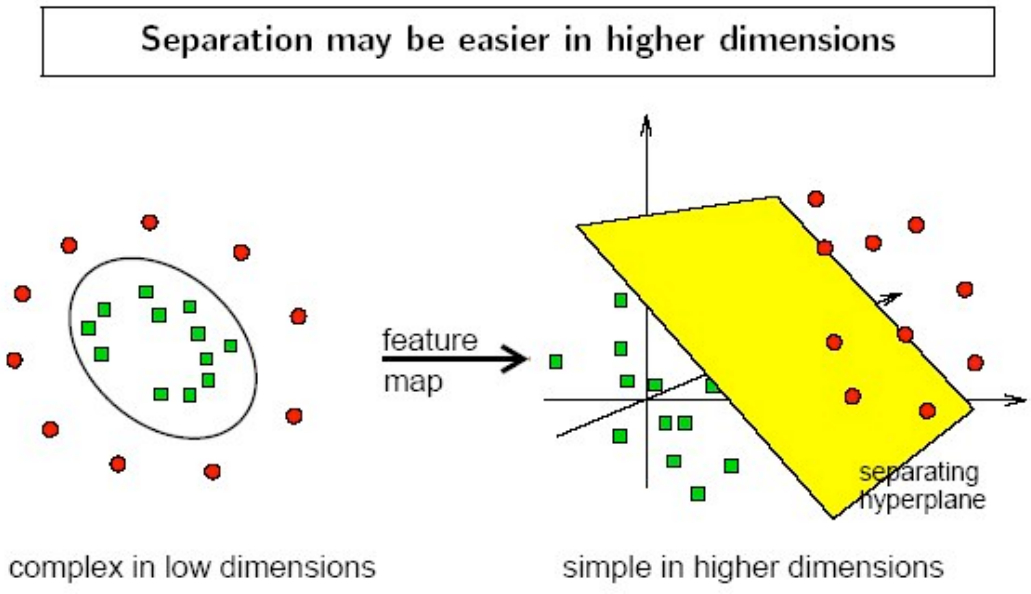


Figura 23. "Ejemplo gráfico del truco del kernel". Extraído de [19]

Las principales funciones de núcleo, en las cuales γ, r, y d determinan los parámetros de los núcleos, son:

- Lineal: $K(x_i, x_j) = x_i^T x_j$.
- Polinomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$.
- Radial Basis Function (*RBF*): $K(x_i, x_j) = exp(-||x_i - x_j||^2), \gamma > 0$.
- Sigmoide: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$.
- Intersección de histogramas: $K(x_i, x_j) = min(x_i^T, x_j)$.

Este último ha resultado ser una buena solución para características que tienen la forma de histogramas normalizados como es el caso de *Bag of words*. En el caso de *Gist descriptor* utilizaremos el *kernel* lineal y el *RBF*.

Escalar los datos antes de aplicar SVM es muy importante [23]. La principal ventaja de escalar, es que se evitan que las características alcancen grandes rangos numéricos y se establezcan pequeños rangos numéricos. Otra ventaja es evitar dificultades numéricas durante el cálculo. Dado que los valores del núcleo, por lo general, dependen de los productos interiores de los vectores de características, los valores grandes de estas podrían causar problemas numéricos. Se recomienda escalar linealmente cada característica en el intervalo [-1, +1] o [0, 1].

2.6 Métodos de evaluación

Para determinar si nuestra solución es la adecuada a nuestro problema debemos de utilizar una serie de medidas de calidad que evaluarán el rendimiento del clasificador, con ello podremos realizar más pruebas y comparar si los cambios efectuados mejoran nuestro clasificador o no. Las medidas que se han utilizado en este proyecto son el *Accuracy* o exactitud, y la AP o *Average Precision*.

2.6.1 Accuracy

Medida de calidad que halla la exactitud de acierto en el clasificador.

$$Acc = \frac{verdaderos\ positivos + verdaderos\ negativos}{total}$$

Ecuación 17. "Ecuación de Accuracy en la clasificación binaria"

El *Accurracy* es a menudo el punto de partida para analizar la calidad de un modelo predictivo. Mide la proporción de predicciones correctas para el número total de los casos evaluados. Puede parecer obvio que la proporción de predicciones correctas a los casos debe ser una buena métrica, pero un modelo predictivo puede tener una gran exactitud y ser ineficiente. Esto sucede en el caso en que tengamos un número de verdaderos negativos mucho mayor que verdaderos positivos o viceversa. Para ello necesitamos otra medida que se pueda ajustarse a un gran número de datos [16].

2.6.2 Average Precision

Para describir esta medida de calidad debemos tener en cuenta otras dos medidas, ya que toman parte en esta medición. Estas medidas son *Precision* y *Recall*, ambas basadas en la proporción de documentos recuperados y documentos relevantes [18]. En la Figura 24 observamos un ejemplo gráfico y descripción para *Precision* y *Recall*, usando diagramas de Venn.

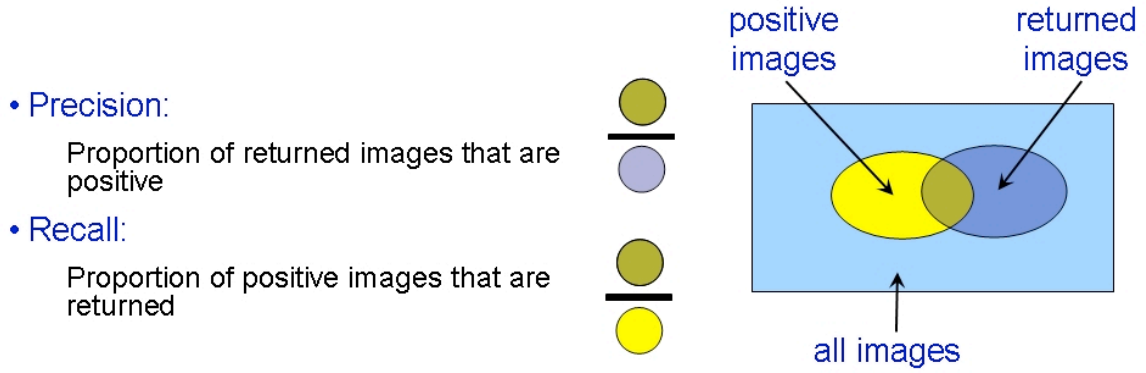


Figura 24. "Explicación gráfica con diagramas de Venn de Precision y Recall". Extraído de [19]

Precision es la fracción de los documentos recuperados que son relevantes a la necesidad de información del usuario.

$$Precision = \frac{|\{documentos\ relevantes\} \cap \{documentos\ recuperados\}|}{|\{documentos\ recuperados\}|}$$

Ecuación 18. "Precision"

Recall es la fracción de los documentos que son relevantes para la consulta que se han recuperado correctamente.

$$Recall = \frac{|\{documentos\ relevantes\} \cap \{documentos\ recuperados\}|}{|\{documentos\ relevantes\}|}$$

Ecuación 19. "Recall"

Recall puede ser definida como la probabilidad de detección de un objeto que es relevante, y la *Precision*, puede ser definida como la probabilidad de detección de un objeto que es recuperado.

La curva de *Precision* y de *Recall* se calcula mediante la variación del umbral en el clasificador (de mayor a menor) y el trazado de los valores de *Precision* contra *Recall* para cada valor del umbral. Con el fin de evaluar el rendimiento de *Recall* en un único número (en lugar de ser una curva), se calcula a menudo el *Average Precision* (AP, que se representa como el área bajo la curva) [20].

Además, *Precision* y *Recall* son medidas basadas en toda la lista de documentos recuperados y relevantes del sistema. Para los sistemas que devuelven una secuencia ordenada de documentos es deseable tener en cuenta también el orden en que los documentos devueltos se presentan. Por ello, el cálculo de *Precision* y *Recall* en cada posición, en la secuencia ordenada de los documentos, se puede trazar como una curva de *Precision* y *Recall*, mostrando la *Precision* como una función de *Recall*, *p(r)*. *Average Precision* se define como la media de *p(r)* en el intervalo de $r = 0$ hasta $r = 1$ [17].

$$\mathrm{AveP} = \int_0^1 p(r)dr$$

Ecuación 20. "Average Precision"

Capítulo 3

Desarrollo del proyecto

3.1 Introducción

Este capítulo se describirá la implementación de la tarea de clasificación de escenas en un programa. Además, contendrá un esquema con el objetivo de resumir por bloques las tareas que se van ejecutando en el programa. También, se hablará de los experimentos realizados para encontrar la solución. Y por último, se darán los resultados del programa a cada posible configuración.

Para llevar a cabo la clasificación se usarán dos técnicas de extracción de características vistas en el capítulo anterior del estado del arte: *gist descriptor* y *bag of words*, y se empleará una *SVM* con diferentes *kernels*. Estos métodos se procederán a evaluar con diferentes medidas de calidad y después, se tomarán las conclusiones a estas técnicas para optar por la solución más eficiente para ser integrado en el sistema de anotado automático.

3.2 Base de datos empleada

Las categorías seleccionadas pertenecen a la base de datos *Fifteen Scene Categories* de *The Ponce Group* [9]. Cada una de las quince categorías presententes en la base de datos tiene de 200 a 400 imágenes, y de media unos tamaños de imagen de 300 píxeles x 250 píxeles. La Figura 25 contiene tres ejemplos de la base de datos. Esta es una de las más completas bases de datos de categorías sobre escenas visuales usadas en la literatura hasta ahora [5]. La base de datos tiene un total de 4.485 imágenes, y se compone de las siguientes categorías:

- **Bedroom**: contiene 216 imágenes de dormitorios.
- **CALsuburb**: contiene 241 imágenes de viviendas en el extrarradio.
- **Industrial**: contiene 311 imágenes de industrias.
- **Kitchen**: contiene 210 imágenes de cocinas.
- **Livingroom**: contiene 289 imágenes de salones.
- **MITcoast**: contiene 360 imágenes de costa.
- **MITforest**: contiene 328 imágenes de bosque.
- **MIThighway**: contiene 260 imágenes de autopistas.
- **MITinsidecity**: contiene 308 imágenes desde dentro de la ciudad.
- **MITmountain**: contiene 374 imágenes de montaña.
- **MITopencountry**: contiene 410 imágenes de campo abierto.
- **MITstreet**: contiene 292 imágenes de calles.
- **MITtallbuilding**: contiene 356 imágenes de grandes edificios.
- **PARoffice**: contiene 215 imágenes de oficinas.
- **Store**: contiene 315 imágenes de supermercados.

Figura 25. "Ejemplos de la base de datos". Extraídas de [9]

3.3 Implementación del proyecto

3.3.1 Esquema del proyecto

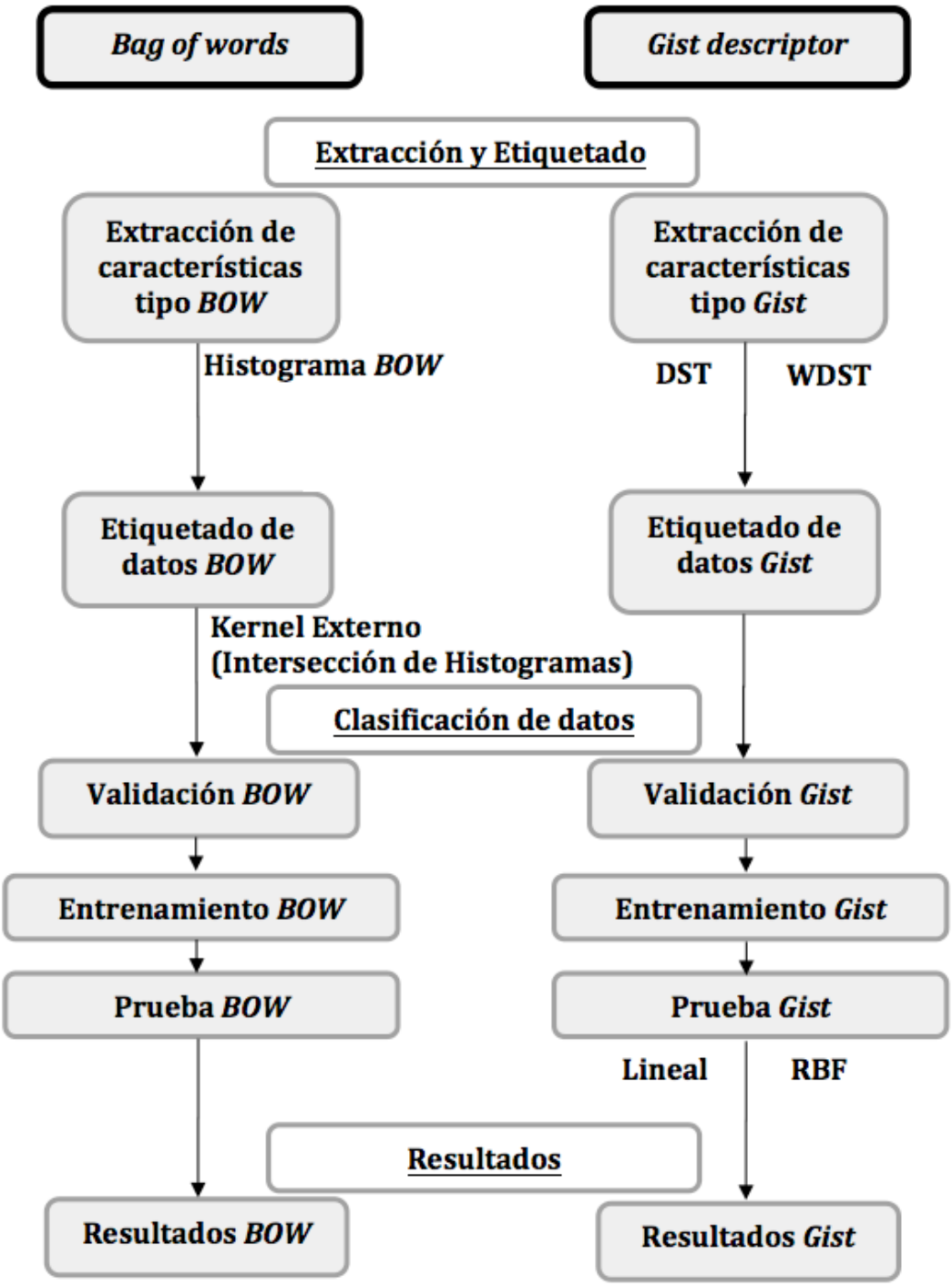


Figura 26. "Esquema del proyecto"

Para ambas implementaciones, *Bag of words* y *Gist Descriptor*, el esquema que se ha utilizado en el proyecto es muy similar (ver Figura 26). Primeramente, se extraen todas las características según el método de extracción deseado para cada imagen, guardando previamente la figura de la transformación realizada. Seguidamente se realiza el histograma normalizado para el caso de *Bag of words*. Después, ambas entran en la fase de etiquetado de los datos para construir los vectores de etiquetas de cada clase. En la fases de clasificación tenemos tres tareas claramente diferenciadas: validación, entrenamiento (train) y evaluación (test). Y por último, se evalúan los resultados para extraer conclusiones del estudio.

3.3.2 Lenguaje de programación

El lenguaje de programación que se ha utilizado es el lenguaje M. Este lenguaje es un lenguaje propio del software matemático *MATLAB,* que incluye además su propio entorno de desarrollo integrado (*IDE*). Se ha decidido usar este lenguaje para la programación del proyecto ya que es un software que es muy usado en las universidades, sobretodo en ingeniería, ciencia y economía. Por tanto, el alumno ya conoce su sintaxis y ha experimentado con el a lo largo de la carrera. Además es un software en el que el número de prestaciones sigue creciendo debido a su gran utilización en investigación y desarrollo. Permite fácilmente la manipulación de matrices, visualizado de funciones y datos, en este caso, de imágenes, creación de interfaces de usuario e implementación de algoritmos. Actualmente se encuentra en la versión *MATLAB* 8 (R2012b).

3.3.3 Librerías externas

Aparte de usar las librerías que implementa *MATLAB*, a lo largo del proyecto se han utilizado una serie de librerías externas para diferentes fines. Su uso, en algunos casos, queda justificado debido a que *MATLAB* no proporciona las funciones necesarias para llevar a cabo una determinada tarea y, en otros casos, porque estas librerías implementan soluciones más optimas que las dadas por *MATLAB*. En la siguiente lista se resume cada librería utilizada, aunque en otras secciones de este capítulo se comentarán en profundidad, cuando su utilización tenga lugar.

A. *Gist Descriptor*

Esta librería se ha utilizado en el proyecto para la extracción de características de tipo *Gist*. Posiblemente es la única *toolbox* que exista para la extracción del *Gist Descriptor* ya que su creador, Antonio Torralba, traslado su investigación, sobre esencia de escenas, directamente a *MATLAB* para su posterior uso. Contiene tres funciones principales para su uso ellas son: LMgist.m, función que hace todo el procesado de *gist*, imresizecrop.m, función auxiliar que permite cuadra las imágenes para su posterior extracción de característica, y por último, showGist.m, la cual permite el visualizado de la característica extraída en LMgist.m. También incluye un *script*, demoGist.m, con el que podremos hacer las primeras pruebas con dos imágenes de demostración que contiene el archivo. Esta librería se puede descargar de [30].

B. *VLFeat*

Esta librería de código abierto implementa algoritmos de visión artificial, tales como *SIFT*, *MSER*, y el algoritmo k-medios jerárquico, entre otros. Está escrito en *C* por eficiencia pero ofrece una interfaz hacia *MATLAB* para la facilidad de uso. Es compatible con *Windows*, *Mac OS X* y *Linux*. La última versión de *VLFeat* es 0.9.16 [7]. De esta librería se usará las funciones vl_sift, vl_plotfram, vl_ikmeanspushy vl_ikmeanshist. Más tarde se detallará su funcionamiento y utilidad en el proyecto.

C. *LibSVM*

LibSVM es un software integrado para la clasificación de máquinas de vector soporte, (C-SVC, nu-SVC), para la regresión mediante funciones (épsilon-SVR, nu-SVR) y para la estimación de distribuciones (SVM monoclase). Es compatible con clasificación multiclase [8]. Incluye los diferentes tipos de *kernel* ya vistos anteriormente, lineal, polinomial, *radial basis function* (*RBF*) y sigmoide. Esta librería tiene soporte para múltiples lenguajes de programación. Contiene svmtrain, para validación y *train*, y svmpredict, para la etapa de *test.*

D. *LS-SVMlab1.8*

Puesto que *LibSVM* permite la inclusión de funciones de núcleo externas, para el caso de *Baf of words* haremos uso de esta capacidad y utilizaremos un *kernel* más apropiado para este fin. La función de núcleo que hemos utilizado recibe el nombre de kernel_matrix.m. Este archivo se ha descargado de la librería LS-SVMlab1.8 [32]. Construye un *kernel* simétrico en una matriz, que debe ser positiva para satisfacer la condición de Mercer. Recibe tres parámetros de entrada: Xtrain, matriz de datos de entrenamiento, distance_type, es el tipo de *kernel*, para nuestro problema utilizaremos 1) *Histogram intersection*, y el último, Xt, es la matriz de datos de test, se utiliza solo en el momento en que vayamos a realizar el test [33].

3.3.4 Extracción de características

A. *Gist Descriptor*

Para llevar a cabo esta tarea, se ha utilizado las funciones de la librería externa de *Gist Descriptor*. En este bloque nos encargaremos de extraer las característica *Gist* para todas las imágenes de la base de datos.

Para comprobar las salidas de *Gist Descriptor* y su código, se han realizado diferentes pruebas con las imágenes de demostración que contenía la librería externa y, para familiarizarnos con este método, se han realizado algunas pruebas con la base de datos que se ha utilizado en el proyecto. En primera instancia, se ha tratado de averiguar qué atributos se han de especificar al parámetro *param* y para qué se utilizan. El parámetro *param* contiene toda la información relativa al tipo de descriptor *Gist* que deseemos. En este proyecto, se ha experimentado con un parámetro en particular del descriptor, *param*, al que se le ha especificado la siguiente opción:

- *numberBlocks*, es el número de ventanas, de largo y de ancho, que tendrá nuestro filtro. Por tanto, para *DST* usaremos una ventana 1x1, por definición, y para *WDST* usaremos 4x4, ya que es el tamaño de ventanas más utilizado en [2].

En el programa del proyecto, se ha recorrido todo el directorio de las distintas categorías, e imagen a imagen se ha ido haciendo la transformación para *DST* y para *WDST*. Cada vector de características estará descrito por un banco de filtros Gabor para 4 escalas en 8 orientaciones, por tanto en el caso de *DST* tendremos una salida de 32 filtros y para *WDST* tenemos 32 filtros para cada bloque, en total 512 dimensiones (32 filtros x 4x4 bloques). Posteriormente, se han almacenado los vectores de *Gist* para cada transformada de todas las imágenes, y se han dividido estos vectores en *test* y *train,* con un 30% y 70% respectivamente. Por último, se han guardado los nombres de estas imágenes para después ser utilizados en el etiquetado de datos.

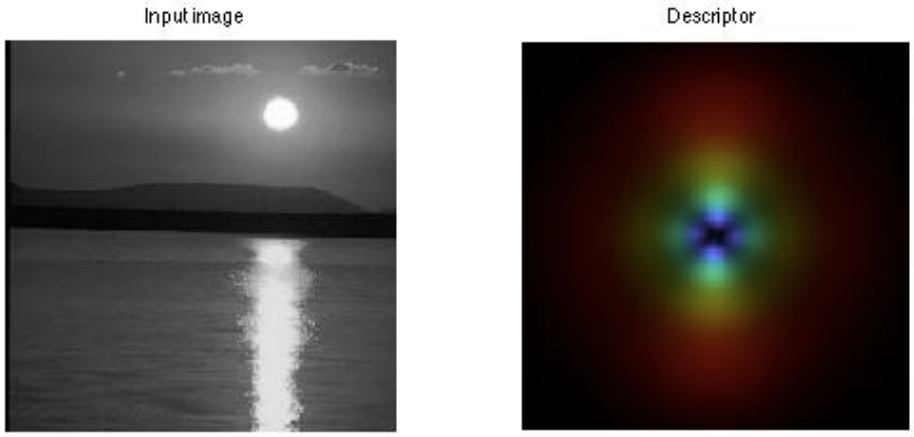


Figura 27. "Ejemplo de extracción de Gist descriptor DST"

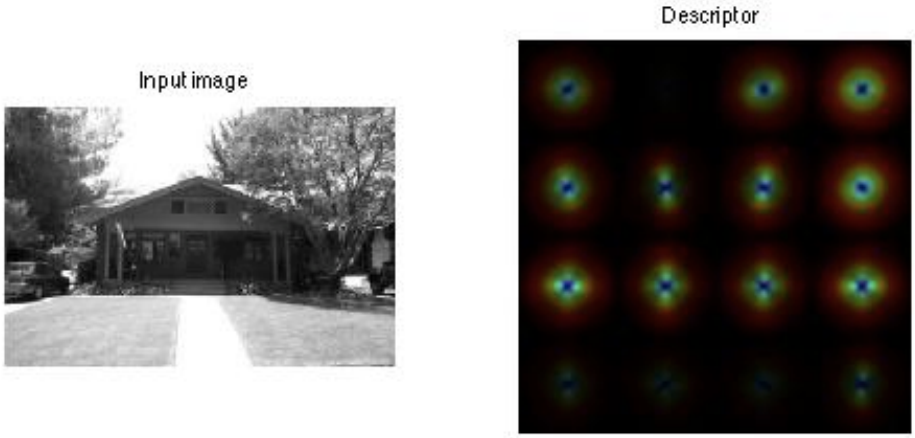


Figura 28. "Ejemplo de extracción de Gist descriptor WDST"

De manera complementaria, en este bloque además se puede visualizar las salidas del *Gist descriptor*, para *DST* (ver Figura 27) y para *WDST* (ver Figura 28). Y además, si el usuario lo desea, podrá almacenar una copia de las figuras que muestran los *Gist Descriptor* de cada imagen.

B. *Bag of words*

La extracción de características de *BOW* se ha realizado con la librería *vl_feat*, descrita en la sección anterior. Esta etapa se ha dividido en dos partes, la extracción de los descriptores de cada imagen y la creación de sus correspondientes histogramas normalizados.

En primer lugar, se extraen todos los detectores y descriptores de tipo *SIFT* para cada imagen. Una vez completado el bucle para una clase de escena se pasa a la siguiente hasta completar el total de clases. Para extraer el vector de descriptores *BOW* de una imagen se ha requerido el uso de detectores de tipo *SIFT*. Estos detectores proporcionan una serie de puntos de interés (*keypoints*) que se identifican a través de su localización, su escala y orientación (Figura 29). Cada *keypoint* se describe después mediante un descriptor *SIFT*, que tienen una dimensión de 128 (Figura 30). En esta fase, como se hacía para *Gist*, también tenemos la opción de visualizar y almacenar las figuras de los descriptores para cada imagen.

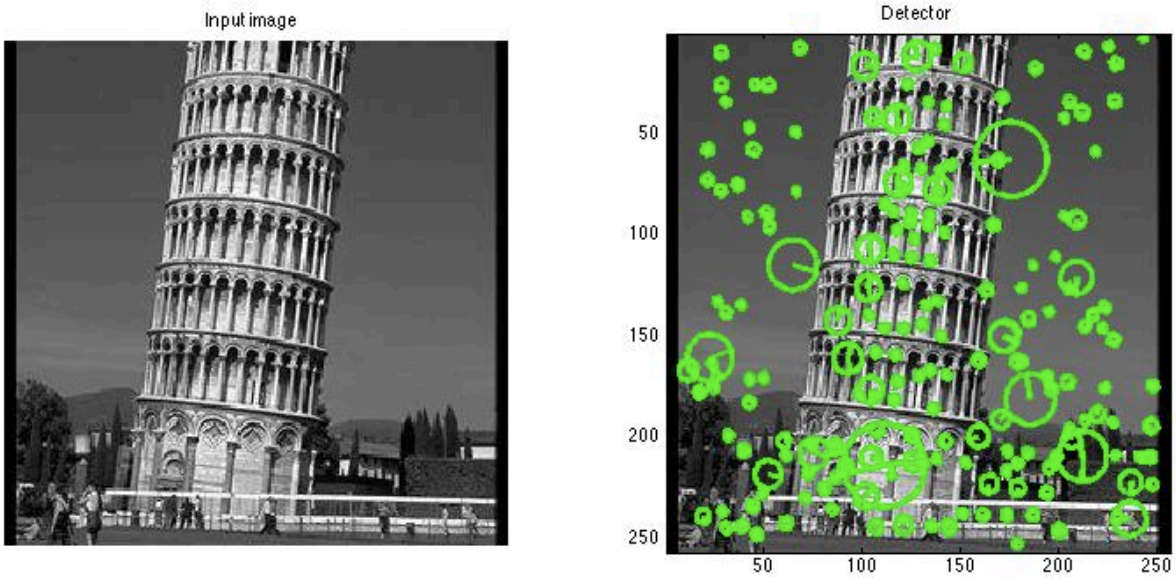


Figura 29. "Ejemplo de extracción del detector SIFT"

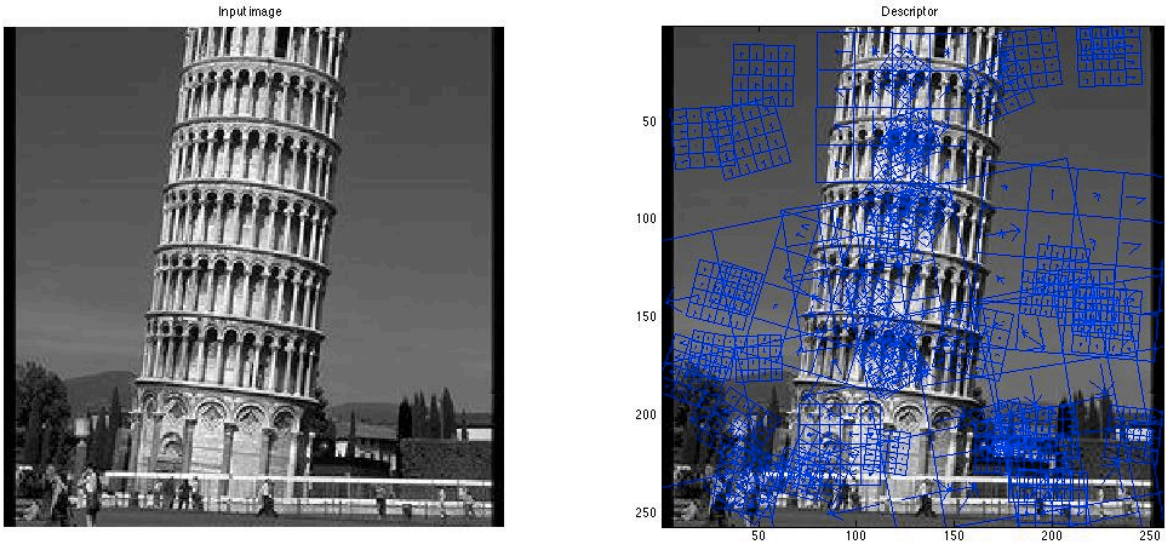


Figura 30. "Ejemplo de extracción del descriptor SIFT"

La segunda tarea para completar la extracción de características consiste en primera instancia, crear un amplio vocabulario de descriptores y luego, asignar cada palabra de las imágenes a una del vocabulario, creando así un histograma. Para nuestro proyecto hemos escogido un número de muestras que se desean extraer para formar el vocabulario igual a 100.000 y el número de centroides o palabras de nuestro vocabulario va a ser 1.000. Este tamaño del vocabulario permite obtener soluciones suficientemente discriminativas con un coste computacional no muy elevado [6].

Primero calculamos el número total de imágenes, sumando los contadores de todas las categorías. Llegados a este punto, se crea un set que contenga varios descriptores de imagen elegidos al azar. El número de descriptores que tendrá va a ser similar al número de muestras que hemos introducido como parámetro. Es similar y no exactamente el mismo porque vamos a escoger un número constante de descriptores por imagen para crear el set, con lo que este número se redondeará, dado que se tiene que coger un número entero de descriptores. El número total de descriptores escogidos al azar es el valor constante de descriptores para cada imagen por el total de las imágenes. En el caso del proyecto, haciendo cálculos, se coge 22 descriptores por imagen (100.000 muestras / 4.485 imágenes = 22,297 descriptores), lo que hace un total de 98.670 palabras en el set (22 descriptores * 4.485 imágenes).

Una vez el set se ha completado se procede a la creación del vocabulario con un algoritmo de agrupamiento del tipo *k-means*, el cual devuelve un vector de los centroides (dimensión de los descriptores x número de centroides) haciendo particiones de los datos. En la función *vl_sift*, que nos proporciona *vl_feat*, se debe de introducir la siguiente opción:

- Méthod: algoritmo que usará ('Lloyd' o 'Elkan'). Valor prefijado, Lloyd.

Considerando el caso que nos abarca, se ha elegido el método Elkan. Ambos métodos originan la misma salida, pero Elkan requiere menos cálculos de distancia, y por lo tanto, menos tiempo de computación. En la Figura 31 se muestra un ejemplo de cómo se crea un vocabulario de 3 palabras a partir del agrupamiento de un conjunto de datos con el algoritmo *k-means*.

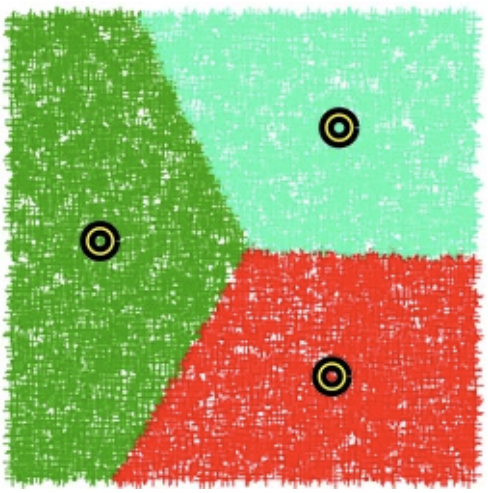


Figura 31. "Ejemplo de agrupamiento de los datos en un plano con un número de centroides igual a 3". Extraída de [7]

Seguidamente, se procederá a iterar en cada tipo de escena para asociarle un histograma al descriptor *SIFT* de cada una de las imágenes. Para este fin, se han de proyectar los descriptores de cada imagen en el plano de los centroides, de modo que cada descriptor se asociará con el centroide más cercano. Este proceso conocido como Cuantificación Vectorial, nos devolverá un vector con el número de índices asignados a cada descriptor, de tamaño igual al número de descriptores en la imagen. Luego se han de calcular los histogramas para cada asignación del paso anterior. Después se debe hacer

una normalización del histograma dividiendo cada valor por el total. Este hecho tiene la ventaja de que cada histograma se podrá modelar como una función de densidad de probabilidad discreta, ya que la suma de los datos es 1. Además, cada valor estará representado por la probabilidad de aparición de esa palabra en toda la imagen.

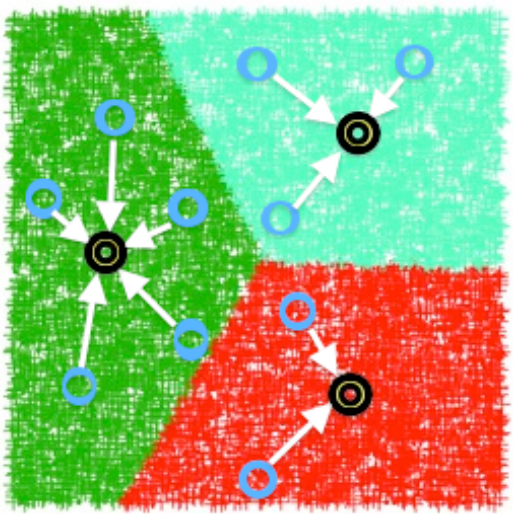


*Figura 32. "*Ejemplo de agregación de los datos en un plano con un número de centroides igual a 3*"*

En la Figura 32 se representan varios datos dispersos por el plano de descriptores, que se asociarán a un centroide, o lo que es lo mismo, a una palabra del vocabulario, de esta forma cada dato se describirá por el centroide más cercano a él.

Finalmente, se almacenará una matriz que contiene el total de histogramas para esa categoría y también un vector de nombres al igual que hicimos con *Gist*.

En las Figuras 33 y 34 se pueden observar un ejemplo del proyecto sobre la normalización del histograma. Se puede comprobar que la estructura es la misma, solo supone un cambio de escala por un factor igual al numero total de valores.

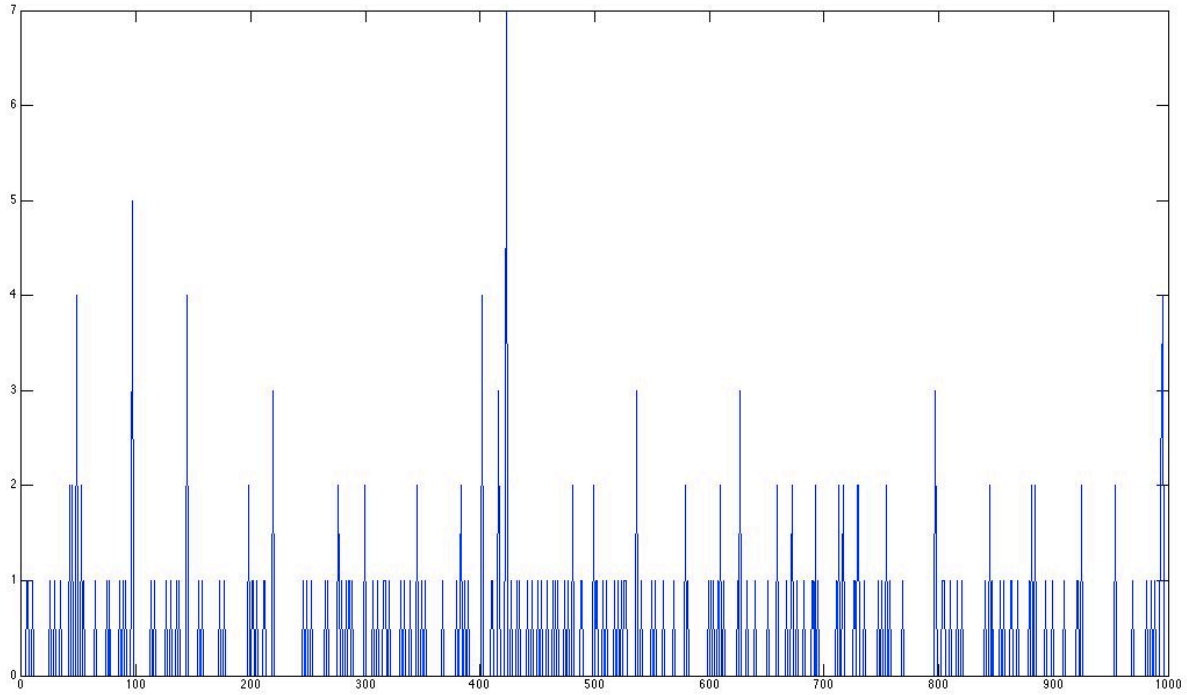


Figura 33. "Ejemplo del proyecto de histograma"

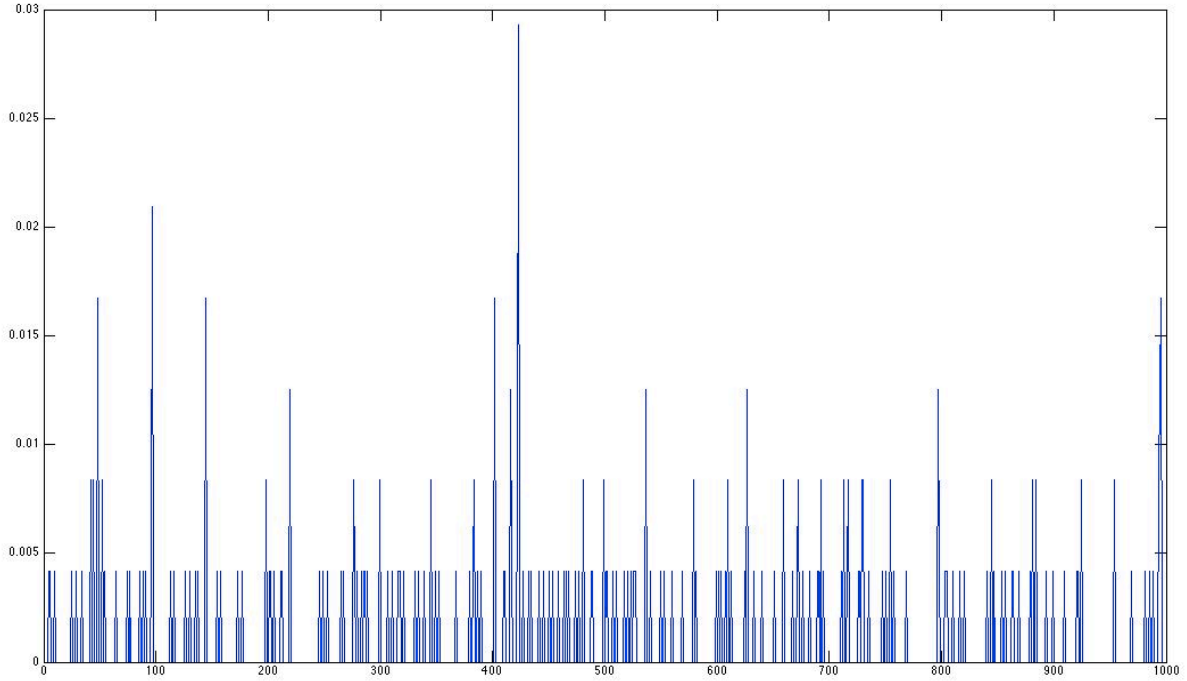


Figura 34. "Ejemplo del proyecto de histograma normalizado"

3.3.5 Etiquetado de datos

La mecánica que se utiliza para etiquetar los datos, para ambos tipos de extracción, *Gist descriptor* y *Bag of words*, es la misma. Primero se cargan la matriz de características, de cada escena, en cada tipo de datos, *train* y *test*, y los vectores de los nombres de las imágenes para cada escena.

Para que nuestro programa no tenga errores a la hora de clasificar, se desordenan las matrices de características, de este modo evitaremos que las características de una determinada escena estén aglutinadas en la matriz. Para cada categoría se genera un vector de etiquetas de *train* y *test* que indique si es o no perteneciente a la clase y la posición en que se encuentra para clasificación binaria. Para ello, se realiza una comparación entre el nombre de una escena almacenado como patrón y cada elemento de la matriz de patrones. Así, si el elemento contiene el nombre de la escena la salida será un 1 y si no será un 0. En la Figura 35 se presenta la explicación de como un patrón que se almacenó en la fase de extracción de manera desordenada se transforma en un vector de etiquetas para un tipo de escenas.

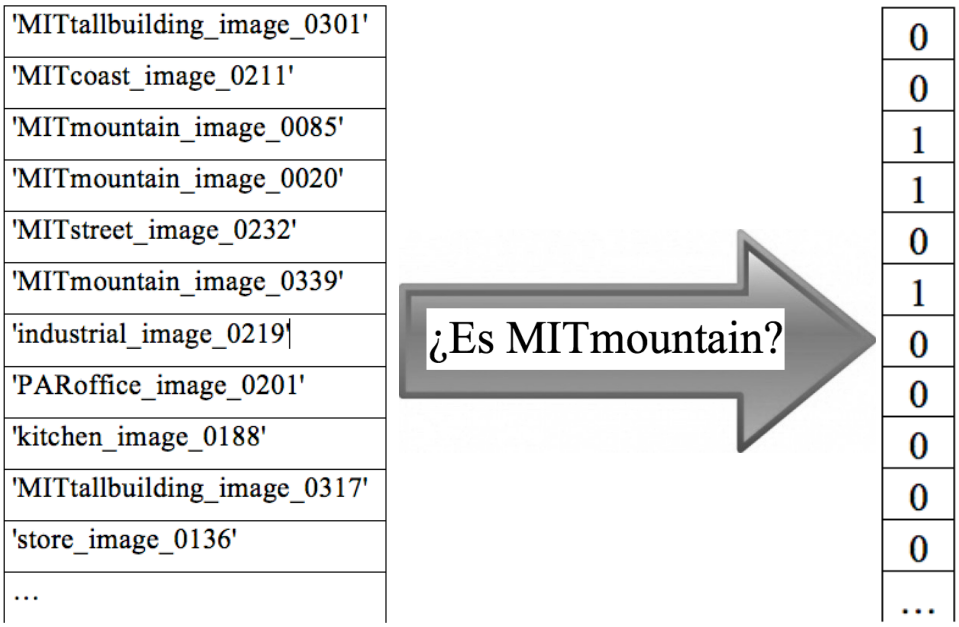


Figura 35. "Explicación de etiquetado de datos"

3.3.6 Validación del clasificador tipo *SVM*

Entrados en la tarea de clasificación, la parte de validación es, según la experiencia aportada por el proyecto, la más costosa computacionalmente y la que más tiempo ha requerido. Es en este periodo donde se trata de buscar los parámetros óptimos para nuestra máquina de vector soporte haciendo uso de una validación cruzada (*Cross-Validation*) [51]. Este tipo de validación es una técnica utilizada en clasificación en la que se divide el grupo de datos de entrenamiento en un número específico *k*, de subconjuntos. Después, se generar *k* conjuntos de validación, cogiendo *k-1* para *train* y el restante para *test*. La clasificación se hará *k* veces en las que cada vez será un subconjunto de *test* diferente y los restantes serán los conjuntos de *train*. Por lo tanto, cada subconjunto debe ser una vez conjunto de *test*. Por último, se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. La Figura 36 contiene un esquema que explica las fases de la validación cruzada con un $k = 4$.

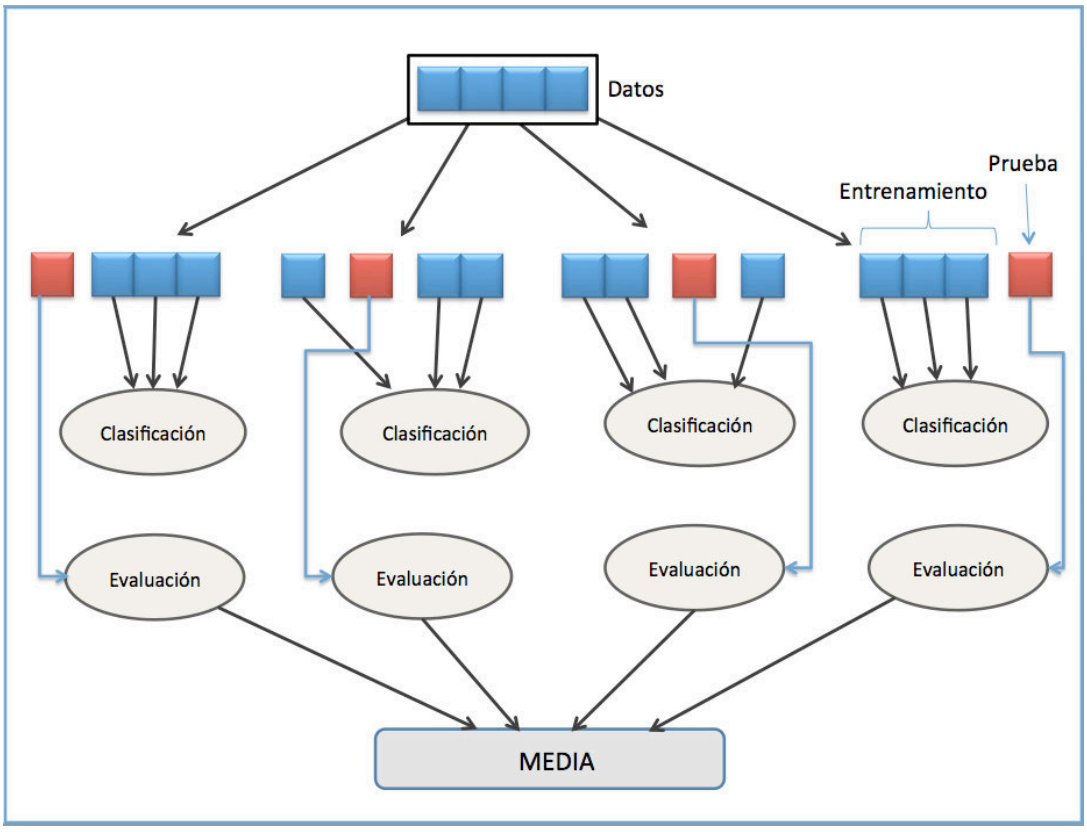


Figura 36. "Esquema de validación cruzada con k = 4". Extraído de [44]

Se escoge un número para *k* según el número de las muestras que tengamos. Si escogemos números muy altos de *k* para un número de muestras reducido, se generarán defectos en la estabilidad estadística ya que los subconjuntos no serán buenas representaciones del conjunto de muestras; por otro lado, si escogemos un valor bajo de *k*, existe el riesgo de sobreajuste en la validación de los parámetros.

En base a la cantidad total de datos que tenemos en train (3.134 características de cada imagen), se escoge un número de $k = 5$. Así, cada subconjunto de validación cruzada tendrá alrededor de 626 datos.

Para llevar a cabo este cometido en el proyecto utilizaremos la librería LibSVM. En la validación cruzada se devuelve el *Accuracy* de este proceso. Es por ello que para elegir el valor óptimo de parámetros debemos utilizar dos estrategias:

- buscar el valor de los parámetros mediante una búsqueda exhaustiva, y escoger el valor que haya tenido mayor *Accuracy*, ó

- un algoritmo en el que se ajuste unos parámetros iniciales y en cada iteración cambie estos parámetros para encontrar los valores óptimos intentando buscar el máximo *Accuracy* (búsqueda por gradiente).

Esta segunda forma es la utilizada en el proyecto, debido que es la que más ahorro de computación y de tiempo supone, mientras que se ha comprobado en alguna prueba preliminar, cómo ha sido capaz de alcanzar el máximo de la validación.

A. *Gist Descriptor*

Para el descriptor de tipo *Gist* se han elegido dos tipos de *kernel* diferentes para realizar la clasificación. El tipo lineal (*linear: u'*v*) y el tipo *radial basis function* o *RBF* (*radial basis function: exp(-gamma*|u-v|^2)*). A la hora de validar un *kernel* lineal, únicamente se tiene que escoger el valor óptimo del parámetro *C* de la ecuación 13. Mientras que en el *RBF*, aparte del valor del parámetro *C*, se necesita validar la variable *gamma* propia del *kernel*. Por tanto, en este caso, el problema será bidimensional.

Primero cargamos las matrices de característica de los datos de entrenamiento para DST y para WDST. Después, se crean unas variables que contendrán los valores óptimos de la validación cruzada para cada tipo de escenas. Estas variables serán finalmente almacenadas para su posterior uso en entrenamiento, además de almacenar una variable con todos los valores de cada parámetro y variable evaluada.

Se necesita asignar los pesos a las muestras positivas y a las negativas, pues el conjunto de datos de entrenamiento es claramente no balanceado, con alrededor de un 90% de datos negativos y un 10% de datos positivos. Esto podría degenerar en soluciones erróneas que proporcionasen todas las salidas nulas con un *Accuracy* alto. Por ejemplo, para una clase de 300 imágenes en las que se muestre que ninguna imagen pertenece a la categoría que se está evaluando nos mostrará un *Accuracy* alrededor de 93%.

El algoritmo utilizado para la búsqueda del *SVM* con *kernel* de tipo lineal contiene los siguientes pasos:

- Inicialización de variables de apoyo en la búsqueda de parámetros, empezamos con *C* = 2^(-2) y con un paso de +4 en el exponente de *C*. Inicialización de la variable que almacenará los valores que ha ido tomando el parámetro *C*.
    - Ejecución de bucle infinito.
    - Ajuste de las opciones en cada iteración. En este caso, el tipo de svm en *C-SVM*, el tipo de kernel en lineal, la validación cruzada en 5 subconjuntos, el ajuste de cada pesos, y por último, el valor de *C* se asigna a un valor en cada iteración.
    - Ejecutar la *SVM*, guardar el valor en la variable de valores para esa categoría y guardar en otra variable el máximo de *Accuracy*.
    - Si se ha alcanzado el máximo en una posición intermedia se habrá encontrado el valor óptimo y sale del bucle. Si el máximo está en uno de los extremos se define un nuevo extremo en esa dirección y se re ejecuta el bucle. Las Figuras 37 y 38 muestran el modo de búsqueda para este caso, X denota el máximo local de *Accuracy* y la flecha la dirección de las siguientes búsquedas.

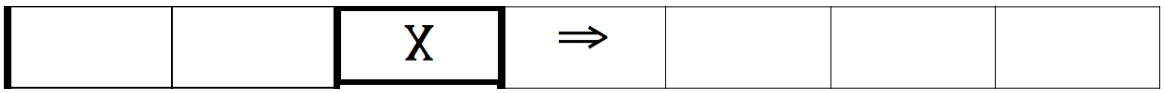


Figura 37. "Modo de búsqueda para el parámetro C máximo"

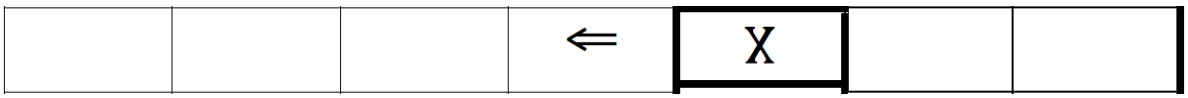


Figura 38. "Modo de búsqueda para el parámetro C mínimo"

El algoritmo utilizado para la búsqueda del *SVM* con *kernel* de tipo *RBF* contiene los siguientes pasos:

- Inicialización de variables de apoyo en la búsqueda de parámetros, empezamos con *C* y *gamma* = 2^(-2) y con un paso de +4 en el exponente de *C y gamma*. Inicialización de la variable que almacenará los valores que ha ido tomando el parámetro *C* y la variable *gamma*.
    - Ejecución de bucle infinito.
    - Ajuste de las opciones en cada iteración. En este caso, el tipo de svm en *C-SVM*, el tipo de kernel en *radial basis function*, la validación cruzada en 5 subconjuntos, el ajuste de cada pesos, y por último, el valor de *C* y de *gamma* se asigna a un valor en cada iteración.
    - Ejecutar la *SVM*, guardar el valor en la variable de valores para esa categoría y guardar en otra variable el máximo de *Accuracy*.
    - Si se ha alcanzado el máximo en una posición intermedia se habrá encontrado el valor óptimo y sale del bucle. Si el máximo está en uno de los extremos se define un nuevo extremo en esa dirección y se re ejecuta el bucle. Las Figuras 39, 40 y 41 muestran el modo de búsqueda para este caso, X denota el máximo local de *Accuracy* y la flecha la dirección de las siguientes búsquedas. La Figura 42 muestra los casos límite en los que se encuentren *C* y *gamma* en un extremo. Si esto ocurre, se realizará la búsqueda dando siempre prioridad al parámetro C. Con un inconveniente, si después de evaluar *C* y proceder a evaluar *gamma* encuentra el máximo en la columna denotada por 0, el algoritmo para.

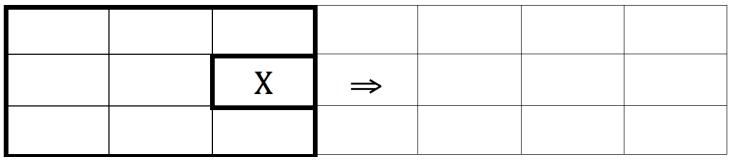


Figura 39. "Modo de búsqueda para el parámetro C máximo"

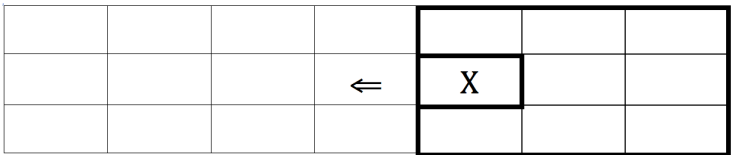


Figura 40. "Modo de búsqueda para el parámetro C mínimo"

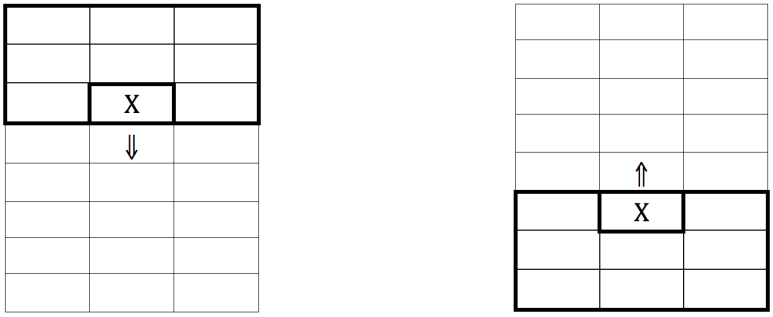


Figura 41. "Modos de búsqueda para la variable gamma máximo y gamma mínimo respectivamente"

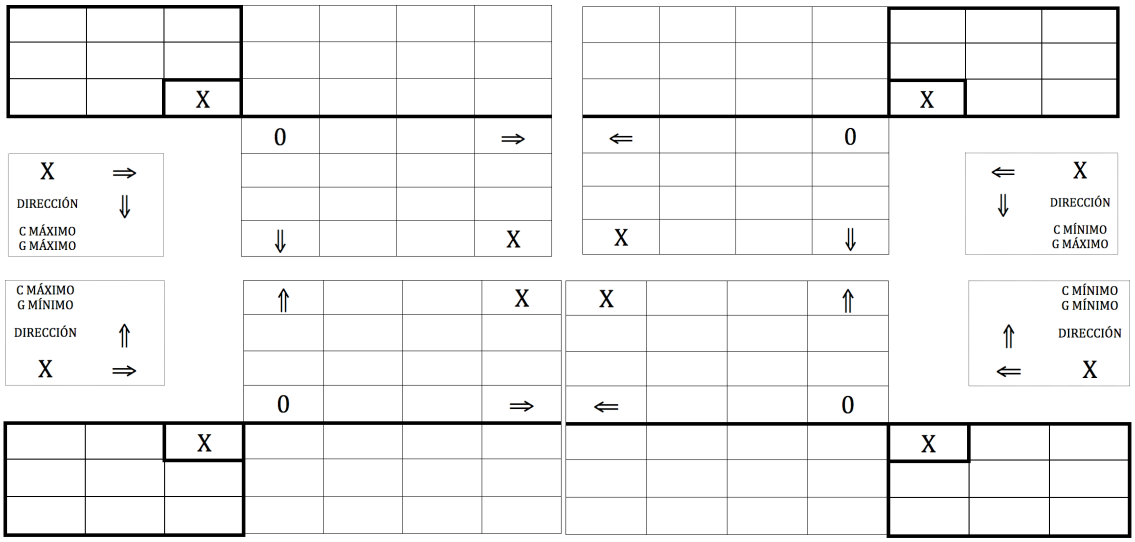


Figura 42. "Modos de búsqueda para el párametro C y la variable gamma"

B. *Bag of words*

Para la validación de *Bag of words* se ha elegido utilizar un tipo de *kernel* más apropiado para este tipo de datos. El *kernel* que se usará es del tipo intersección de histogramas.

Al igual que en el caso de *Gist*, cargamos las matrices de característica de los datos de entrenamiento, creamos la variable para almacenar los valores óptimos para cada escenas, y, al final, las almacenaremos para usarlas en entrenamiento junto con la variable de todos los valores ejecutados de la validación cruzada.

El algoritmo utilizado para la búsqueda del *SVM*, para *Bag of words,* con el *kernel* externo sigue los mismos pasos que en el caso lineal para *Gist*, a excepción de la inclusión del kernel en la *SVM*. Para ello, se ajustarán las opciones de la *SVM* con el tipo de *kernel* externo. Dado que *LibSVM* no incorpora este *kernel* se empleará una función externa de la librería LS-SVMlab1.8, que se introducirá como tipo de kernel el *Histogram Intersection*.

3.3.7 Entrenamiento del clasificador tipo *SVM*

Después de encontrar los valores óptimos para los parámetros y variables de las distintas *SVM*, se pasa a la fase de entrenamiento. Su funcionamiento es el mismo para los dos casos de extracción de características. Al igual que en la tarea de validación, se especifican las opciones para el clasificador de cada tipo de escena. En este caso debemos usar las mismas opciones que utilizamos en la validación de SVM; tipo de SVM, tipo de *kernel* y pesos según la etiqueta, pero en este caso no especificamos la opción de validación cruzada y damos a las variables y parámetros el valor óptimo que obtuvimos. De esta forma tendremos un modelo de clasificación para cada tipo de escena, que serán almacenados para comprobar su eficacia en la fase de evaluación.

3.3.8 Evaluación del clasificador tipo *SVM*

En ambos casos, *Gist descriptor* y *Bag of words*, se procede a iterar en cada categoría para evaluar cada modelo de clasificación. La tarea devuelve el *Accuracy* del proceso, es decir las imágenes que ha clasificado correctamente, el vector de las etiquetas que ha predicho (las cuales muestran 1 o 0 según pertenezcan o no a la categoría), y son comparados con un TH = 0 para generar la salida dura de clasificación, y la salida blanda del decisor, es decir, aquellos valores numéricos que devuelve el clasificador para cada dato. Cuanto más se acerque la salida blanda a menos infinito más seguridad tenemos de que ese dato no pertenezca a la escena evaluada y, cuanto más se acerquen a infinito más certeza tenemos de que es un dato de nuestra categoría.

Las salidas de la evaluación para cada escena, vector de etiquetas predichas, *Accuracy* y la salida blanda, serán guardadas para *Gist descriptor* y *Bag of words*.

3.3.9 Resultados del clasificador tipo *SVM*

Por último, se procede a evaluar la clasificación hecha para cada categoría. Este bloque nos devolverá una matriz de resultados con un resumen de las medidas de calidad de los modelos para cada escena y la media de cada método de extracción y tipo de *SVM*. Para cada categoría, se calculará el *Average Precision*, puesto que es una medida más adecuado que el *Accuracy* para problemas de recuperación de información. Al final de este proceso debemos tener las medidas de calidad del *SVM* lineal y *SVM RBF* para las transformada *DST* y *WDST*, en *Gist descriptor*, y la de *Bag of words* con el *kernel* de *intersección de histogramas*.

3.4 Resultados

3.4.1 Validación de parámetros

En el caso de la validación la principal preocupación es encontrar los parámetros óptimos de la *SVM*. Para ello debemos seleccionar correctamente los valores iniciales y el paso del algoritmo para que no converja a un máximo local únicamente. Haciendo pruebas se ha decidido empezar por un valor de *C* y *gamma* igual a 2^(-2) y un paso de +4 en el exponente con saltos de 0,5. En las siguientes tablas se mostrarán los valores en Accuracy y el valor óptimo de los parámetros encontrados.

A. *Gist descriptor*

Para *DST* con *SVM* lineal se han encontrado los siguientes parámetros óptimos:

'scenes'	**'mejorAccLineal'**	**'mejorCLineal'**
'CALsuburb'	58,90	-2
'MITcoast'	92,75	17
'MITforest'	93,13	3,5
'MIThighway'	87,84	9,5
'MITinsidecity'	88,60	11,5
'MITmountain'	88,70	17,5
'MITopencountry'	80,12	13,5
'MITstreet'	76,06	-2
'MITtallbuilding'	89,02	5,5
'PARoffice'	90,87	15,5
'bedroom'	79,41	17,5
'industrial'	69,84	3
'kitchen'	78,36	11,5
'livingroom'	82,64	18
'store'	85,35	9,5

Tabla 1. "Resultados de validación para DST con SVM lineal"

Para *DST* con *SVM RBF* se han encontrado los siguientes parámetros óptimos:

'scenes'	**'mejorAccRBF'**	**'mejorCRBF'**	**'mejorGRBF'**
'CALsuburb'	98,37	15,5	4
'MITcoast'	96,10	19	2,5
'MITforest'	97,38	10,5	6
'MIThighway'	97,16	19,5	2
'MITinsidecity'	91,95	11	1
'MITmountain'	95,18	17	4
'MITopencountry'	93,39	16	4,5
'MITstreet'	76,06	-2	-2
'MITtallbuilding'	96,29	16	4
'PARoffice'	96,45	19,5	4
'bedroom'	92,21	24,5	3
'industrial'	93,80	22	2
'kitchen'	93,36	26,5	2,5
'livingroom'	93,10	24	2
'store'	94,09	21,5	3,5

Tabla 2. "Resultados de validación para DST con SVM RBF"

Para *WDST* con *SVM* lineal se han encontrado los siguientes parámetros óptimos:

'scenes'	**'mejorAccLineal'**	**'mejorCLineal'**
'CALsuburb'	98,62	7,5
'MITcoast'	94,99	9
'MITforest'	96,29	3,5
'MIThighway'	96,49	11
'MITinsidecity'	89,66	1,5
'MITmountain'	94,83	10
'MITopencountry'	89,34	13
'MITstreet'	97,63	9
'MITtallbuilding'	96,36	8,5
'PARoffice'	96,39	9,5
'bedroom'	93,84	13
'industrial'	90,52	13
'kitchen'	92,62	14
'livingroom'	91,89	9,5
'store'	87,58	1,5

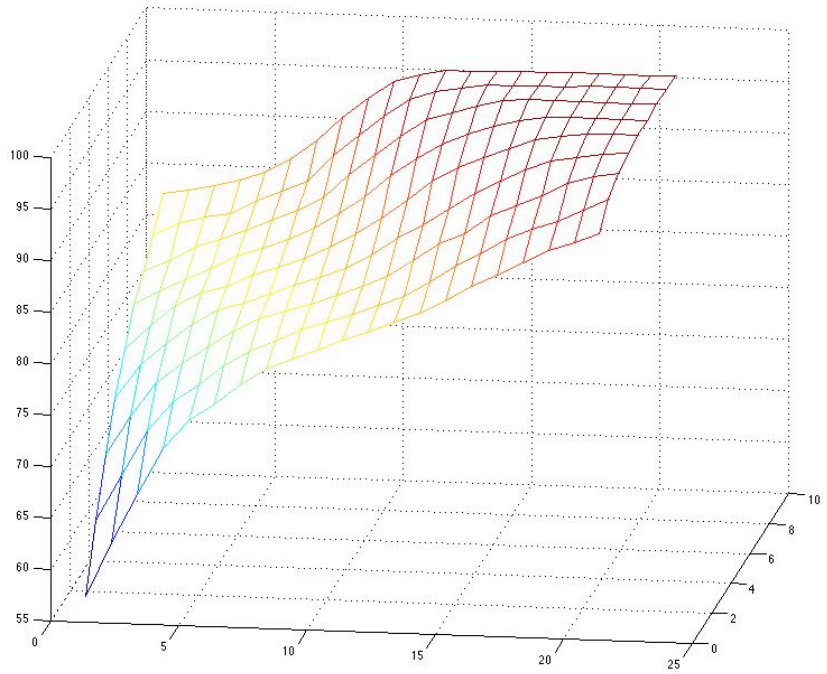
Tabla 3. "Resultados de validación para WDST con SVM lineal"

Para *WDST* con *SVM RBF* se han encontrado los siguientes parámetros óptimos:

'scenes'	'mejorAccRBF'	'mejorCRBF'	'mejorGRBF'
'CALsuburb'	99,61	4	1
'MITcoast'	96,71	5,5	2
'MITforest'	98,43	2,5	2
'MIThighway'	98,50	4,5	1,5
'MITinsidecity'	97,03	6	1
'MITmountain'	97,22	4	1,5
'MITopencountry'	95,05	6	1,5
'MITstreet'	98,43	3	1
'MITtallbuilding'	97,98	3,5	1,5
'PARoffice'	97,35	3	2
'bedroom'	96,10	5,5	1,5
'industrial'	95,30	5	0,5
'kitchen'	96,87	6	1
'livingroom'	96,07	5,5	2
'store'	95,62	3	2

Tabla 4. "Resultados de validación para WDST con SVM RBF"

La Figura 43 muestra un ejemplo de la validación de parámetros *C* y *gamma*. Se observa cómo el algoritmo va buscando la solución óptima para converger en un máximo.



Fi*gura 43. "Ejemplo gráfico de la validación para los parámetros C y gamma"*

B. *Bag of words*

Para *Bag of words* con *kernel* de *intersección de histogramas* se han encontrado los siguientes parámetros óptimos:

'scenes'	**'mejorAccKernelExterno'**	**'mejorCKernelExterno'**
'CALsuburb'	96,74	2,5
'MITcoast'	93,96	4,5
'MITforest'	97,47	2,5
'MIThighway'	96,52	5
'MITinsidecity'	93,96	3,5
'MITmountain'	93,26	3,5
'MITopencountry'	89,66	4
'MITstreet'	93,29	5,5
'MITtallbuilding'	93,10	3
'PARoffice'	95,59	6
'bedroom'	94,47	5
'industrial'	91,22	6,5
'kitchen'	95,14	6
'livingroom'	92,62	6
'store'	92,08	4,5

Tabla 5. "Resultados de validación para BoW"

3.4.2 Comparativas

A. *DST* vs *WDST* y Lineal vs *RBF*

Los resultados del clasificador *SVM* se especifican en *Average Precision*. Como se ha comentado con anterioridad, *AP* fusiona *Recall* y *Precision* de modo que se tiene en cuenta la propia distribución de los datos para los positivos y para los negativos. Por ello, se utilizará el *Average Precision*, el cual muestra unos porcentajes que tienen en cuenta esta deficiencia.

La siguiente tabla muestran el resultado final obtenido para la clasificación de imágenes de la base de datos con la extracción de tipo *Gist*.

'Resultados'	*'CALsuburb'*	*'MITcoast'*	*'MITforest'*	*'MIThighway'*	*'MITinsidecity'*
'DST Lineal'	14,46	86,40	87,57	32,89	57,50
'DST RBF'	85,94	81,11	90,02	61,03	72,23
'WDST Lineal'	90,52	89,36	90,27	59,25	63,54
'WDST RBF'	96,30	89,78	91,36	77,56	80,81

'Resultados'	*'MITmountain'*	*'MITopencountry'*	*'MITstreet'*	*'MITtallbuilding'*	*'PARoffice'*
'DST Lineal'	64,77	46,62	12,56	55,58	55,57
'DST RBF'	78,18	76,24	12,60	63,88	63,35
'WDST Lineal'	76,35	50,46	81,83	74,51	53,15
'WDST RBF'	92,52	81,47	86,70	83,40	65,65

'Resultados'	*'bedroom'*	*'industrial'*	*'kitchen'*	*'livingroom'*	*'store'*
'DST Lineal'	24,08	27,21	21,59	43,72	53,15
'DST RBF'	24,91	69,33	29,34	61,45	47,76
'WDST Lineal'	43,06	43,43	36,48	53,38	60,60
'WDST RBF'	63,72	69,35	50,51	75,07	72,21

Tabla 6. "Resultados finales para Gist descriptor de la clasificación de test"

Podemos destacar que, en reglas generales, se cumple que WDST es mejor que DST y la elección del la *SVM RBF* es mejor que la de tipo lineal. Existen excepciones: en 'MITcoast' y 'store' dan mejores salidas en *DST* con un tipo de *SVM* lineal que con uno de tipo *RBF*, concretamente un 5,29% y un 5,38%, respectivamente. A su vez, para 'PARoffice', se tiene mejores resultados en *DST* que en *WDST* para un *SVM* lineal, 2,42% un mejores.

Se pueden extraer varias conclusiones de la tabla. Evaluando entre *DST* y *WDST* destaca el caso de 'MITstreet', en el que para *DST* da bajos porcentajes, cerca del 12%, y para *WDST* da buenos resultados, ascienden a mas del 80%. Observando las figuras extraídas para esta categoría se ve claramente que para una imagen cualquiera de esta categoría, la estructura de cada celda del *WDST* difiere en gran medida de la del *DST*. Esto implica que la estructura espacial no es constante a lo largo de la imagen, por lo que el *DST* es una representación incompleta. Esto se debe a que la diferenciación espacial en

las imágenes de 'MITstreet' es muy fuerte. La mayoría de imágenes de esta categoría son fotografías donde predomina una gran carretera o calle en la parte baja de la imagen y edificios con muchos ventanales en la parte superior (ver Figura 44).

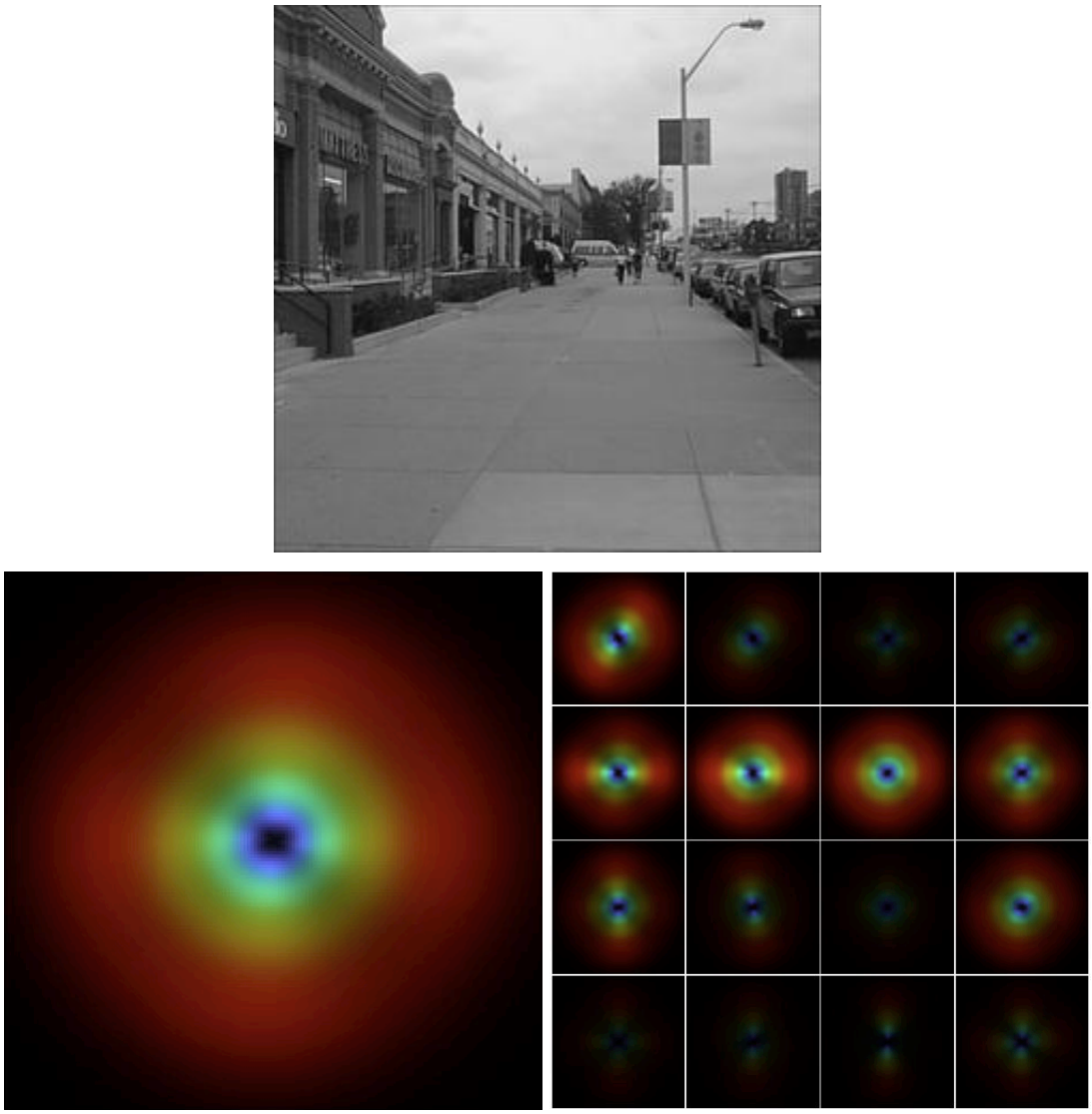


Figura 44. "Comparativa DST vs WDST para una imagen de MITstreet"

Sin embargo, para otras categorías como 'MITforest', en las que la diferenciación en porcentajes de *DST* y *WDST* es muy baja, se comprueba que las celdas de la *WDST* son casi todas iguales y, a su vez, muy parecidas al *DST*. En este otro caso, las imágenes de 'MITforest' predominan los arboles frondosos y arbustos, por lo cual no existe una gran diferenciación espacial en toda la imagen, y por tanto, *DST* será muy parecido a *WDST* como muestra la Figura 45.

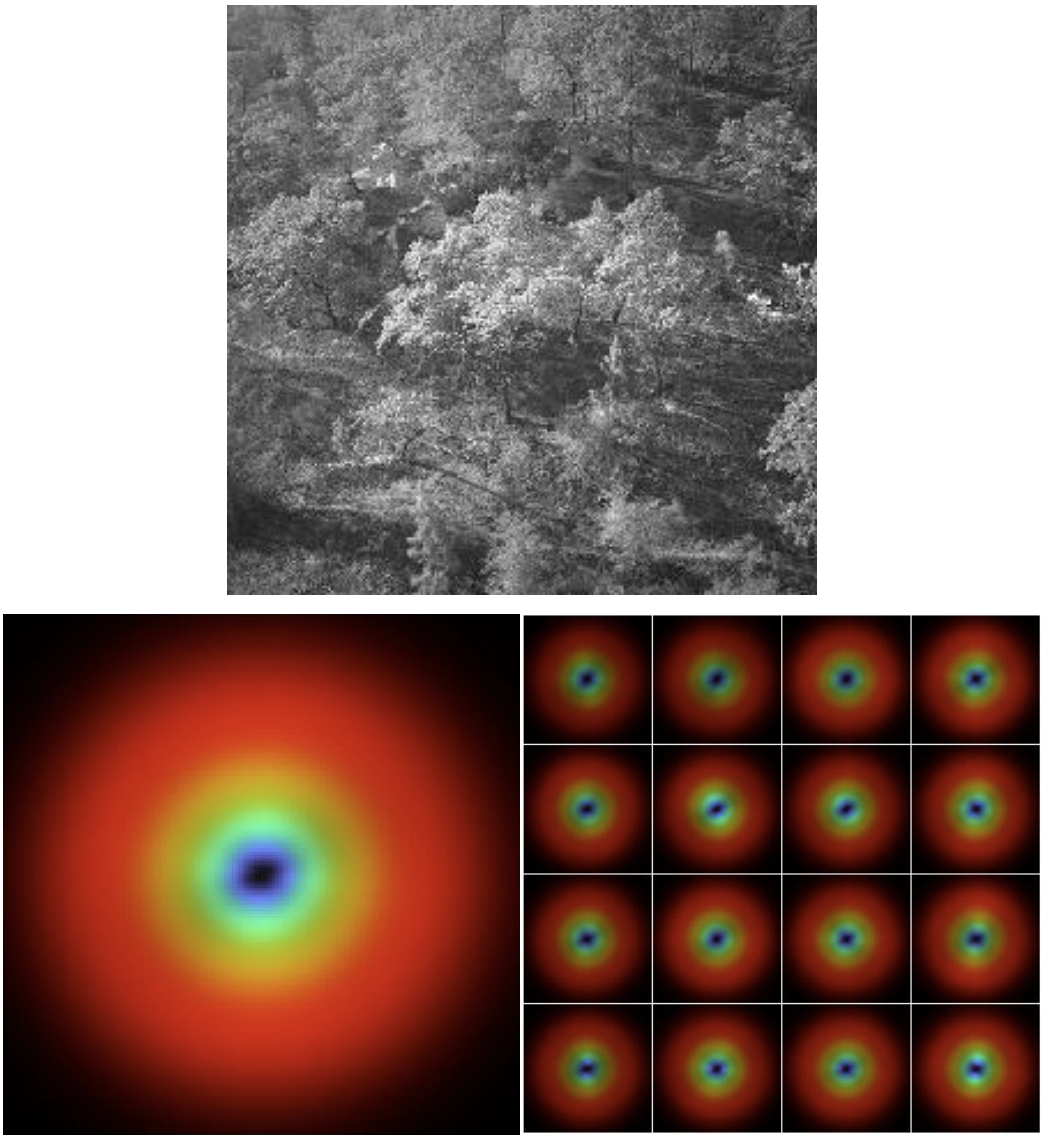


Figura 45. "Comparativa DST vs WDST para una imagen de MITforest"

En media los resultados obtenidos para cada tipo de extracción, *DST* y *WDST*, y para los tipos de *SVM*, *Lineal* y *Radial Basis Function*, son los siguientes:

'Resultados'	*'Average'*
'DST Lineal'	45,58
'DST RBF'	61,16
'WDST Lineal'	64,41
'WDST RBF'	78,43

Tabla 7. "Medias para cada categoría de los resultados finales en Gist"

Observamos que la mejor clasificación la realiza *Windowed Discriminant Spectral Template* con una *SVM* con *kernel RBF* alcanza un *AP* 80% de efectividad. Esto cumple satisfactoriamente los objetivos propuestos ya que estamos trabajando con 15 categorías y *Gist descriptor* estaba optimizado para 8 categorías.

B. *Gist descriptor* vs *Bag of words*

Las siguientes tablas muestran el resultado final obtenido para la clasificación de imágenes de la base de datos con la extracción de tipo *Bag of words*.

'Resultados'	*'CALsuburb'*	*'MITcoast'*	*'MITforest'*	*'MIThighway'*	*'MITinsidecity'*
'DST Lineal'	14,46	86,40	87,57	32,89	57,50
'DST RBF'	85,94	81,11	90,02	61,03	72,23
'WDST Lineal'	90,52	89,36	90,27	59,25	63,54
'WDST RBF'	96,30	89,78	91,36	77,56	80,81
'Kernel IH'	78,20	68,81	83,19	31,94	46,24

'Resultados'	*'MITmountain'*	*'MITopencountry'*	*'MITstreet'*	*'MITtallbuilding'*	*'PARoffice'*
'DST Lineal'	64,77	46,62	12,56	55,58	55,57
'DST RBF'	78,18	76,24	12,60	63,88	63,35
'WDST Lineal'	76,35	50,46	81,83	74,51	53,15
'WDST RBF'	92,52	81,47	86,70	83,40	65,65
'Kernel IH'	56,53	53,98	28,43	46,21	51,51

'Resultados'	*'bedroom'*	*'industrial'*	*'kitchen'*	*'livingroom'*	*'store'*
'DST Lineal'	24,08	27,21	21,59	43,72	53,15
'DST RBF'	24,91	69,33	29,34	61,45	47,76
'WDST Lineal'	43,06	43,43	36,48	53,38	60,60
'WDST RBF'	63,72	69,35	50,51	75,07	72,21
'Kernel IH'	24,34	17,09	29,93	31,56	40,38

Tabla 8. "Resultados finales para Bag of words de la clasificación de test"

'Resultados'	*'Average'*
'DST Lineal'	45,58
'DST RBF'	61,16
'WDST Lineal'	64,41
'WDST RBF'	78,43
'Kernel IH'	45,89

Tabla 9. "Medias para cada categoría de los resultados finales en BoW"

Este resultado nos demuestra que para el reconocimiento de escenas *Bag of words* no da buenos resultados.

Un inconveniente que causa el mal funcionamiento de *Bag of words* es la longitud del vocabulario empleado para generar los histogramas. Se han escogido 1.000 palabras para formar el vocabulario y, como los descriptores totales para una imagen media son alrededor de 300, el resultado de BoW es un histograma normalizado de carácter *sparse*; es decir, con la gran mayoría de sus posiciones a cero (ver Figuras 33 y 34). Este tipo de vectores no proporciona buenos resultados al ser utilizados con *SVM* y *kernel* de intersección de histogramas. Para solucionarlo se podría aplicar la técnica del *Dense SIFT* [46], que aplica una malla a toda la imagen y extrae los descriptores de cada punto de la malla, con lo cual esto nos daría más descriptores por imagen lo que resultaría en un histograma más suave. El problema de esta aplicación es que se eleva mucho el coste computacional, ya que a más descriptores más tiempo de procesado.

Otro problema que tiene *Bag of words* es que no realiza discriminación espacial al igual que *DST*, y como se ha discutido anteriormente, esto es perjudicial para los tipos de escenas escogidos. Esta es la razón principal de porque los resultados obtenidos para *BoW* y *DST* son muy parecidos, y en ambos casos, peores que *WDST*. Para solventar este problema se puede usar *Spatial Pyramid Matching*, que es una extensión de *BoW* y hace diferenciación en el espacio. Por esto, va a ser una de las propuestas en las líneas a seguir después de este proyecto.

C. Coste computacional

Se procederá a evaluar los tiempos en los que se tarda en realizar la extracción de una imagen para los dos modelos propuestos. Para tener una idea de los tiempos que se tarda en la ejecución de los hilos principales, para todo el proceso (validación, entrenamiento y evaluación) de *Gist descriptor* se ha tardado exactamente 211.101,36 segundos (próximo a las 59 horas), y para *Bag of words* se tardan 4.062,48 segundos (1 hora y 10 minutos). Para tener medidas los más aproximadas a todo el conjunto se va a realizar un promedio para unas cuantas imágenes escogidas aleatoriamente.

- Gist descriptor:

  El tiempo medio medido para la extracción del *DST* de una imagen incluyendo la creación y asignación de valores al parámetro *param* es de: 0,71 segundos.

  El tiempo medio medido para la extracción del *WDST* de una imagen incluyendo la creación y asignación de valores al parámetro *param* es de: 0,75 segundos.

Dado que las dimensiones del vector de características *WDST* es NxN veces más grande que el de *DST*, siendo N el número de bloques especificado en *param* (para nuestro caso *WDST* tiene N = 4). Por ello, es normal que el tiempo de computación de *WDST* sea superior al de *DST*.

- Bag of words:

  El tiempo medio medido para la extracción del descriptor SIFT es de: 0,067 segundos.

  El tiempo medio medido para la asignación de palabras del vocabulario es de: 0,042 segundos.

  El tiempo medio medido para la creación del histograma normalizado es de: 0,0005 segundos.

El total de estos tiempos suma 0,11 segundos, tiempo mucho menor que el de *Gist*. Este tiempo varia en proporción a los puntos de interés que el detector *SIFT* encuentra en cada imagen. Por ejemplo una imagen con bajo grado de aspereza tardara menos que una con alto grado.

D. Resultados visuales

Finalmente, se muestran unos resultados visuales en las siguientes Figuras (46, 47 y 48) para distintas categorías, en las que la primera es la imagen que mejor ha clasificado, la segunda la primera imagen, no perteneciente a la categoría evaluada, que se ha dado como positiva (falsa alarma) y el la primera imagen, perteneciente a la categoría evaluada, que se ha dado como negativo (pérdida).

Figura 46. "Resultados visuales para DST con SVM Lineal en 'MITmountain'"

Figura 47. "Resultados visuales para WDST con SVM RBF en 'MIThighway'"

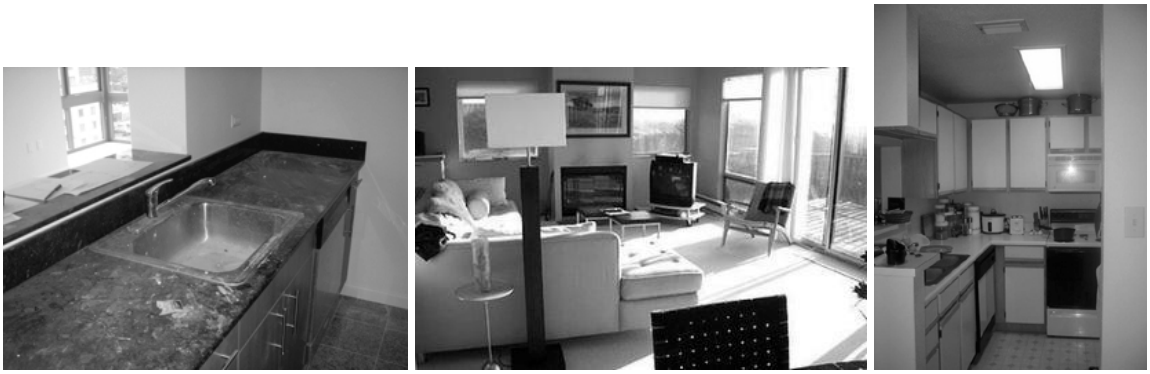


Figura 48. "Resultados visuales para BoW en 'kitchen'"

E. Conclusiones del estudio

En base a las comparativas realizadas se va a escoger *Gist descriptor* en vez de *Bag of words*, porque *BoW* tiene bastantes deficiencias que deben de ser corregidas y *Gist descriptor* hace buena diferenciación de las categorías propuestas. Se ha escogido *WDST* frente a *DST* porque hace diferenciación espacial y, pese a su elevada dimensionalidad, muestra buenos resultados. Y como tipo de *SVM* se ha elegido *RBF* en vez de Lineal, ya que ajusta mucho mejor las Máquinas de Vector Soporte, dado que no estamos en un problema linealmente separable.

En cuanto a tiempos de computación *Bag of words* tiene mejores resultados que *Gist descriptor* pero, aunque sea superior, es asumible para nuestra aplicación porque no tenemos requisitos de tiempo real. Por tanto, como este factor es menos importante que los demás, se ha decidido que finalmente se escogerá *Gist Descriptor* con un número de bloques 4x4, *Windowed Discriminant Spectral Template*, para la extracción de características y, para la clasificación se utilizará la *SVM* con el *kernel* de tipo *Radial Basis Function*.

3.5 Integración en el sistema

A continuación, se va a comentar la estrategia seguida para integrar los modelos en un sistema de anotado de imágenes automático. Para esta última fase del proyecto, vamos a coger varios bloques de nuestro programa para integrarlos en el sistema.

El primero de ellos es la carga de los modelos de las *SVM* para el tipo de *kernel radial basis function*. Después, se integrará la fase de extracción de la característica elegida, en este caso *Gist descriptor*, donde podremos extraer el vector de características para una imagen y dar la opción para el número de bloques que forman *Windowed Discriminant Spectral Template*. El siguiente paso es la clasificación de la imagen para cada uno de los distintos tipos de escenas con el vector de características de la imagen. Finalmente, este proceso nos dará la salida blanda de la clasificación para cada categoría. Si este valor tiende a ser de un valor positivo, nuestra imagen pertenecerá a dicha categoría, y si este valor es negativo, no pertenecerá a esta escena, como ya se explico anteriormente. Puede darse el caso que la imagen pertenezca más de una categoría en ese caso comprobaremos cual es el valor más alto de ellos, esta será la categoría que mejor se ajuste a nuestra imagen.

Capítulo 4

Conclusiones y líneas futuras

4.1 Introducción

Este capítulo mostrará las conclusiones extraídas después de la realización del proyecto. Contará las ventajas e inconvenientes que se han encontrado en los diferentes métodos de extracción, así como en las técnicas utilizadas para la clasificación. Por último, se comentarán las posibles caminos a seguir después de la realización del proyecto, y posteriores pasos para continuar con la investigación respecto al reconocimiento de escenas en imágenes.

4.2 Conclusiones

Para el caso de la extracción de *Gist descriptor*, Oliva y Torralba en [2] habían realizado sus pruebas para 8 categorías diferentes de escenas. En este proyecto se ha probado con casi el doble de categorías, 15. Ha sido una manera tentativa de probar su eficacia para 7 nuevas categorías de las cuales los autores no habían entrenado sus atributos del *Spatial Envelope*. En el proyecto hemos comprobado que estas nuevas categorías se ajustan bastante bien a los atributos de naturaleza, apertura, aspereza, expansión y rugosidad, en algunos casos dando mejores resultados incluso que en las anteriores categorías. Además, observamos que, en líneas generales, se cumple que es mejor la extracción por el modo enventanado, *WDST*, que la extracción de la esencia de la imagen global, *DST*.

Para el caso de *Bag of words*, se comprueba que realiza buenos resultados, pero realmente no cumple las expectativas necesarias que se requieren para la integración en el sistema de anotado automático de escenas. Estos problemas se deben a dos razones principales *BoW* nos muestra histogramas con carácter *sparse* (la mayoría de valores a 0) y no realizada discriminación espacial, por lo que en algunas categorías muestra claras deficiencias.

Las ventajas que presenta *Gist descriptor* son que es invariante frente a transformaciones de luminancia, faltas de definición, como pueden ser imágenes desenfocadas o altamente comprimidas, y además soporta cambios de tamaño en la imagen. Y sus desventajas son que no es invariante frente a rotaciones o translaciones de imágenes, ocultamiento parcial de objetos e imágenes recortadas.

Las ventajas que presenta *Bag of words* son principalmente, que es invariante frente a la posición y la orientación del objeto, tiene unos buenos resultados en la clasificación de acuerdo con los objetos que contiene, y da buenos resultados para las imágenes que tiene un fondo desordenado (útil para algunos tipos de escenas). En contrapartida, es pobre en la localización de objetos dentro de una imagen.

En cuanto a la clasificación se ha escogido el algoritmo de las máquinas de vector soporte, haciendo uso de tres de sus posibles configuraciones, lineal y *radial basis function* para *Gist*, e intersección de histograma para *Bag of words*. Globalmente, RBF tiene mayores tiempos y carga de computación que la configuración lineal, debido a que se tienen que evaluar dos variables, pero nos muestra mejores resultados para los casos de *DST* y *WDST*. En la intersección de histogramas para *Bag of words* hemos podido comprobar que los tiempos y la carga computacional son muy buenos pero no es la solución adecuada para el problema que nos abarca.

Finalmente, se ha escogido la solución de la extracción de *Gist descriptor,* concretamente, la transformada *Windowed Discriminant Spectral Template*, con una *SVM* del tipo *Radial Basis Function*, por ser la solución que mejores valores se han extraído en los resultados para los distintos tipos de escenas evaluadas. Por tanto, se ha decidido que va a ser la candidata perfecta para la integración de nuestro sistema de anotado automático de imágenes.

También se ha completa un objetivo adicional, el aprendizaje del autor. Con este proyecto se ha conseguido que el estudiante asuma toda responsabilidad, siempre con el apoyo y los consejos del tutor. Uno de los alicientes de este tipo de trabajos es que permite al estudiante enfrentarse con problemas reales con los que deberá lidiar en su futura carrera profesional, además de aplicar muchos de los conceptos teóricos y prácticos aprendidos durante su formación en la universidad. Por lo que ha permitido al estudiante obtener una visión global y nuevos conocimientos sobre un lenguaje de programación muy utilizado en ingeniería, y ciertamente, le será muy útil en el futuro. También le ha proporcionado un escenario perfecto, en el que el estudiante ha podido enfrentarse con un lenguaje de programación que ya conocía en los estudios cursados, para realizar el proyecto desde cero, contando con la orientación proporcionada por el tutor del proyecto.

4.3 Líneas futuras

Para ampliar la investigación desarrollada y poder ser incluida en un sistema de anotado automático adecuada para soluciones doméstico o comerciales, deberíamos ampliar la base de datos. Primero necesitaríamos más categorías para evaluar, que abarquen, en dimensiones reales, la mayoría de tipos de escenas que nos podremos encontrar. Además, necesitaremos un mayor número de imágenes por categoría, estas deberían tener más variedad de tamaño, diferentes formatos y resoluciones, y que tengan diferentes transformaciones como rotaciones, translaciones, desenfoque, etc, para ajustar lo máximo posible a un escenario real.

Para mejorar el modelo de *Bag of words* tendríamos que crear un nuevo bloque en el que pudiésemos seleccionar las palabras óptimas para cada categoría, y no una cantidad variable y selección aleatoria, para después ser integradas en el vocabulario. Este hecho haría que nuestros resultados mejorasen de manera considerable. También se podrían probar otros tipos de *kernel* para *Bag of words*, así como el *Spatial Pyramid Matching* [47], que es un método que calcula el histograma en cada una de las partes de la imagen. Representa localmente histogramas en varios niveles de resolución (ver Figura 49). Una de sus principales ventajas es que tienen capacidad de capturar las características co-existentes y proporciona resultados precisos en menos tiempo de computación con una *SVM* de tipo lineal.

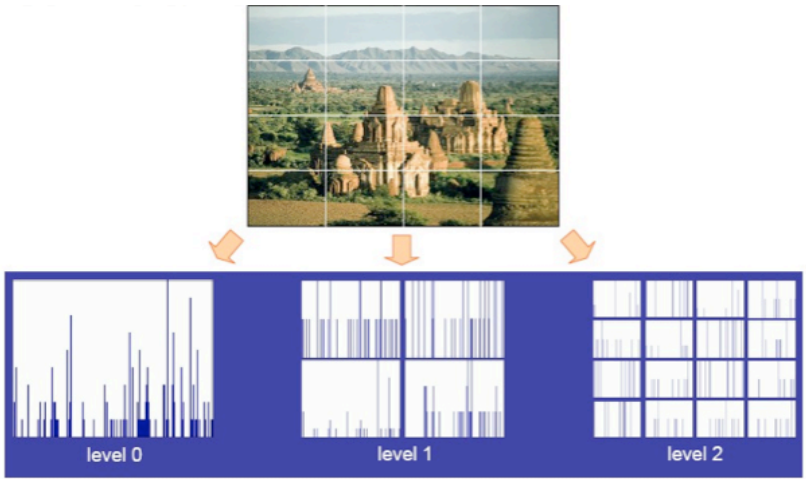


Figura 49. "Ejemplo de Spatial Pyramid Matching". Extraído de [5]

Y para optimizar los resultados de *Gist descriptor* se podría volver a evaluar los atributos que componen el *Spatial Envelope* de [2], para ver si para este grupo de categorías escogida sería mejor tener otro tipo de atributos. Para ello, deberíamos hacer varias pruebas teóricas de clasificación de las categorías, y con respecto a estas pruebas, evaluar un nuevo vector ***d***, asociado a estos atributos, para tener diferentes pesos en las bases *KLT* de *Discriminant Spectral Template* y de *Windowed Discriminant Spectral Template*. Además, habría que hacer un extenso estudio sobre el número de ventanas que se deberían coger para *WDST* en función de las categorías seleccionadas.

Otros métodos más actuales que se podrían utilizar son:

- El aprendizaje de características a medio nivel [50], que utiliza las técnicas de codificación dispersa y la elección del valor máximo (*max pooling*).

- La mejora de *kernel* de Fisher para imágenes de gran escala [48], el cual utiliza descriptores más discriminativos, normalización de la potencia y *SVM* de tipo lineal.

- *Kernels* aditivos y eficientes a través de mapas de características [49], que es una aproximación a través de *kernels* lineales.

Capítulo 5

Presupuesto

5.1 Introducción

Este capítulo incluirá un pequeño desglose del proyecto en fases y subfases con el correspondiente diagrama de Gantt (herramienta de ayuda a la planificación del desarrollo de un proyecto que ilustra un esquema de los tiempos y relaciones entre las distantes actividades que toman cargo en un proyecto) y un presupuesto final que presentará los costes de personal, costes del material y costes totales.

5.2 Esquema de las fases del proyecto

En la Tabla se presentarán las fases del proyecto y las tareas que han tenido lugar en dichas fases. Se añade la fecha de inicio y la fecha de finalización de cada tarea y, además, un breve resumen de cada tarea.

Nombre de la tarea	Fecha de inicio	Fecha de finalización	Información
Fase 1: Lectura y Software			
Lecturas sobre el modelo *Bag of words*	04/05/12	28/05/12	*Papers y memorias de proyectos relacionados con Bag of words*
Repaso de programación en *MATLAB*	28/05/12	05/06/12	*Toma de contacto con el lenguaje de programación*
Lecturas sobre el modelo *Gist descriptor*	05/06/12	06/07/12	*Papers sobre Gist descriptor*
Descarga de las librerías y la base de imágenes	06/07/12	07/07/12	*Búsqueda y descarga de las librerías y bases de datos a utilizar*
Fase 2: Gist descriptor			
Programación de la extracción de *Gist descriptor*	07/07/12	16/07/12	*Aplicación de la extracción de Gist descriptor en bucle a todas las imágenes*
Puebas de *Gist descriptor*	16/07/12	25/07/12	*Comprobaciones de las extracciones de imagen*
Programación del clasificador libsvm Gist para validación	25/07/12	28/08/12	*Búsqueda de parámetros óptimos para las SVM en Gist descriptor*
Programación del clasificador libsvm Gist para train	28/08/12	04/09/12	*Ajuste de los parámetros óptimos a las SVM de entrenamiento*
Programación del clasificador libsvm Gist para test	28/08/12	04/09/12	*Utilización de las SVM sobre los datos de evaluación*
Comprobación de resultados Gist	04/09/12	13/09/12	*Toma de resultados con las medidas de calidad de las SVM para Gist descriptor*

Fase 3: Bag of words			
Programación de la extracción de *Bag of words*	13/09/12	18/09/12	*Aplicación de la extracción de Bag of words en bucle a todas las imágenes*
Puebas de *Bag of words*	18/09/12	21/09/12	*Comprobaciones de las extracciones de imagen*
Programación del clasificador libsvm BOW para validación	21/09/12	25/09/12	*Búsqueda de parámetros óptimos para las SVM en Bag of words*
Programación del clasificador libsvm BOW para train	21/09/12	25/09/12	*Ajuste de los parámetros óptimos a las SVM de entrenamiento*
Programación del clasificador libsvm BOW para test	21/09/12	25/09/12	*Utilización de las SVM sobre los datos de evaluación*
Comprobación de resultados Gist	25/09/12	29/09/12	*Toma de resultados con las medidas de calidad de las SVM para Bag of words*
Fase 4: Integración y memoria			
Solución a problemas generales	29/09/12	31/10/12	*Pruebas realizadas para mejorar errores que se han encontrado debido a malos resultados en las medidas de calidad*
Integración en sistema	31/10/12	04/11/12	*Utilización de la solución más adaptada a nuestro problema y posterior inclusión en el sistema de imágenes a clasificar*
Redacción de la memoria del proyecto	15/10/12	10/11/12	*Búsquedas de referencias, toma de gráficos, redacción general del proyecto…*

Tabla 10. "Tabla de las fases del proyecto y sus correspondientes tareas"

5.3 Diagrama de Gantt

En el siguiente Diagrama de Gantt (Figura 50) podremos ver el desglose de las tareas realizadas en el proyecto, así como sus fechas de inicio y conclusión y su duración. Esta útil herramienta gráfica fue realizada con el programa *GanttProject 2.5.5* para MacOS X de uso libre, por tanto no se ha necesitado licencia del producto.

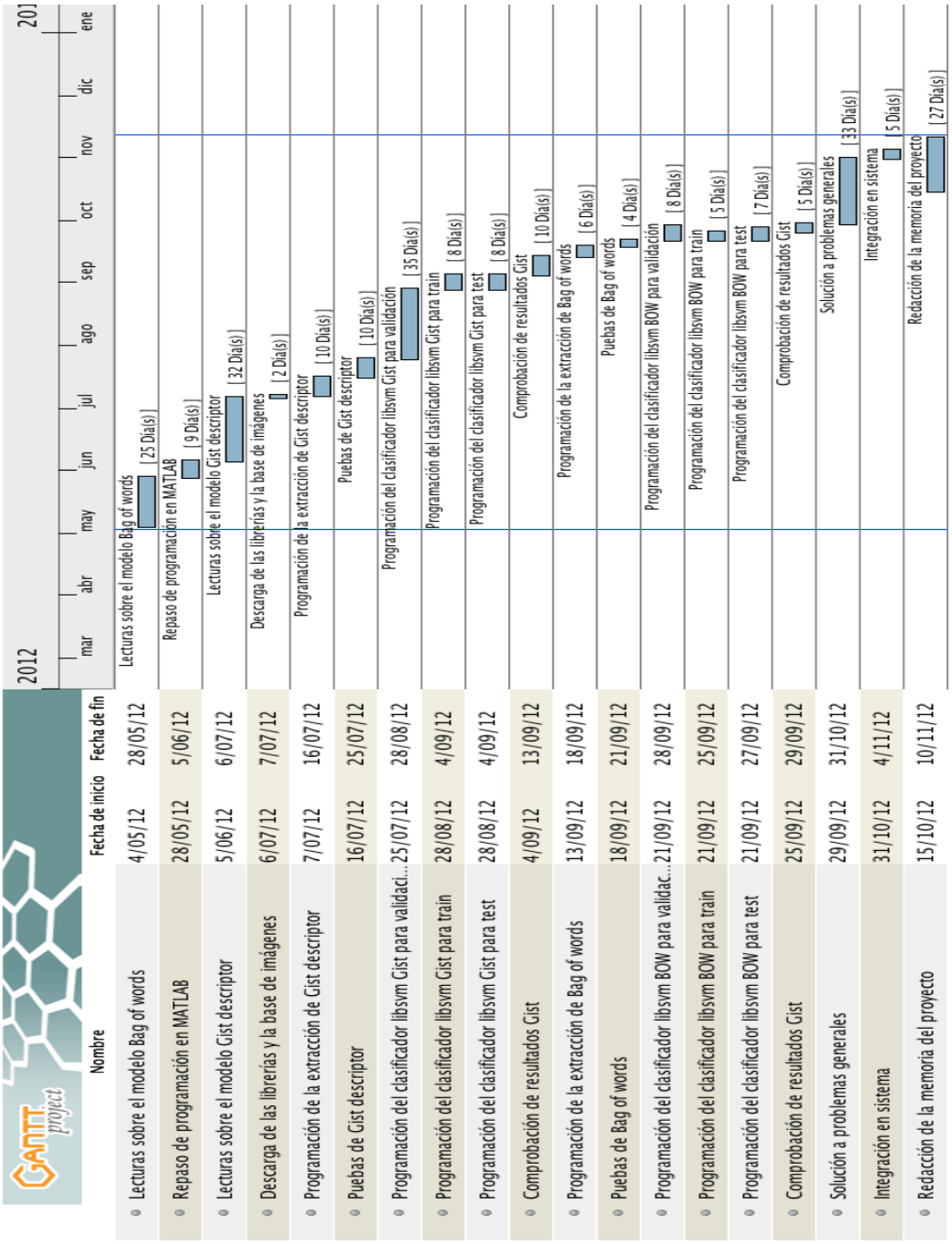


Figura 50. "Diagrama Gantt del proyecto"

5.4 Presupuesto final

5.4.1 Coste material

En los costes de materiales se describe el precio de todas aquellas herramientas que se han necesitado para la realización del proyecto. En la siguiente lista se presentan los materiales utilizados para la realización del proyecto:

- El espacio de trabajo que se ha utilizado es una oficina de alquiler condicionada para el trabajo de pequeñas empresas del sector tecnológico. Esta oficina dispone de toda la tecnología necesaria para llevar a cabo el proyecto, así como electricidad, línea de voz y de datos, calefacción, aire acondicionado y mantenimiento. Todos estos elementos van incluidos en el precio mensual que asciende a 550€, por lo tanto como la duración del proyecto ha sido de 7 meses, el total del espacio de trabajo son **3.850€**.

- El ordenador con el que se ha realizado el desarrollo el proyecto y la redacción de la memoria ha sido un ordenador personal portátil *MacBook* con sistema operativo Mac OS X versión 10.6.8 a 2.4Ghz de velocidad de la CPU. Esta herramienta tuvo un precio de 1.200€, pero dado que su capacidad de reutilización es amplia, suponemos un tiempo útil del aparato de 5 años. Como el ordenador se ha utilizado 7 meses, con una simple regla de tres, tenemos que el precio del ordenador es de **140€**.

- El programa utilizado para la realización del proyecto ha sido *MATLAB*, por lo tanto necesitaremos una licencia de software *MATLAB* para Mac OS X, que tiene un precio de 2.150$, aproximadamente 1.700€. Como este producto al igual que el ordenador se puede amortizar en 5 años, el tiempo de vida del ordenador, en los 7 meses de duración obtenemos un precio final de **200€**.

- También se necesitó la licencia de software Microsoft Office 2011 para Mac OS X, ya que hemos utilizado Word para la redacción de la memoria del proyecto, y Excel para la visualización de las variables que hemos ido guardando en *MATLAB*. Esta licencia cuesta **119€**.

- Por último, hay que considerar una serie de consumibles que se han tenido que utilizar durante el proyecto. En el siguiente listado se muestran los materiales que se han tenido que comprar, ellos son:
    - 100 folios a 0,05 céntimos de euro el folio son **5€**.
    - 3 bolígrafos tipo *Pilot* que han costado **4€**.
    - 2 cartuchos de tinta para la impresión de memorias y papers **60€**.
    - Impresión y encuadernado de la memoria para los miembros del tribunal, para el tutor y el alumno, **150€** (30€ cada copia).
    - El total de los materiales de oficina asciende a **219€**.

5.4.2 Honorarios

Para calcular los costes personales asociados al proyecto se debe tener en cuenta las horas de trabajo realizadas en él, así como al total de personas que han estado a cargo del proyecto.

En primer lugar, se declaran los costes asociados a la dirección del proyecto que ha estado a cargo de Iván González Díaz, que ha tenido un total de 70 horas de dedicación en el proyecto como tutor del mismo, y dado que el coste de la hora laboral de un ingeniero en telecomunicación senior es de 72€, tenemos que el coste de dirección asciende a:

$$70\ horas\ \frac{72\ €}{hora} = 5.040€$$

Figura 51. "Coste de dirección"

Y la segunda persona que ha realizado el proyecto y la redacción de la memoria es el estudiante Víctor Suárez Paniagua. El susodicho ha hecho este proyecto en un periodo de 7 meses, en el que han transcurrido 173 días, descontando fines de semana, en la que se ha tenido una dedicación diaria de 4 horas. Teniendo como referencia que una hora de trabajo tiene un coste de 36€ para ingenieros de telecomunicación junior, se tiene que los gastos asociados a la realización del proyecto son:

$$173\ días\ \frac{4\ horas}{día}\ \frac{36\ €}{hora} = 24.912€$$

Figura 52. "Coste de realización"

5.4.3 Presupuesto total

Finalmente haciendo un resumen de todos los gastos que ha supuesto el proyecto, tenemos que:

Tipo de gasto	Precio en €
Espacio de trabajo	3.850
Ordenador	140
Licencia MATLAB	200
Licencia Office 2011	119
Material de oficina	219
Coste de dirección	5.040
Coste de realización	24.912
TOTAL	**34.480**

Tabla 11. "Presupuesto total"

"El presupuesto total de este proyecto asciende a la cantidad de TREINTA Y CUATRO MIL CUATROCIENTOS OCHENTA EUROS.

Leganés a 10 de noviembre de 2012 El ingeniero proyectista

Fdo. Víctor Suárez Paniagua"

Capítulo 6

Anexo

6.1 Introducción

En los siguientes anexos se incluirán todos los archivos programados en el proyecto, su organización en directorios, descripción de funciones auxiliares y la integración al sistema de anotado automático. Unas breves aclaraciones sobre la mecánica del programa nos aclararan su utilización en caso de futuras líneas de investigación, o, como ya se ha comentado, si se desean clasificar otro determinado número de escenas con este mismo programa para cualquier otro tipo de aplicaciones relacionada con clasificación.

6.2 Mecánica del programa

Lo primero de todo se debe aclarar la distribución de los archivos de la base de datos que tiene que tener el usuario de nuestro programa. Debido a su naturaleza generalista, el programa puede ser extendido a más categorías que las que contiene la base de datos con las que se han hecho las pruebas. Por lo tanto, conviene dar una serie de instrucciones para los usuarios que quieran realizar sus propias pruebas con otra serie de categorías y, así, poder evaluar los resultados obtenidos en base a estas nuevas clases.

En primer lugar, la base de datos se ha de colocar en una carpeta, y dentro de esta, las subcarpetas serán nuestras clases a clasificar. Los nombres de estas subcarpetas deberán ser lo más apropiadas a la categoría, para que la lectura de las variables y resultados a lo largo del programa sea lo mas clara posible. Después de este paso, debemos guardar la ruta del directorio y definir una variable de *MATLAB* de tipo cadena de caracteres con el nombre DIR. Si se desea, también se puede guardar los subdirectorios en la carpeta principal del programa, en ese caso no hará falta definir el DIR, puesto que el programa interpretará que las carpetas de las clases están en el directorio actual. Si bien solo desea cotejar los resultados, tan solo tiene que descargarse la base de datos, y guardar la ruta en la variable DIR, puesto que esta base de imágenes ya guarda la estructura adecuada (Figura 53).

Es recomendable agregar al *path* del entorno de desarrollo *MATLAB* las librerías externas y toda la librería del programa del proyecto. Para ello, vamos a la etiqueta File > Set Path… y agregamos los directorios necesarios con Add Folder… o en su caso Add with Subfolders… si queremos agregar las subcarpetas del directorio.

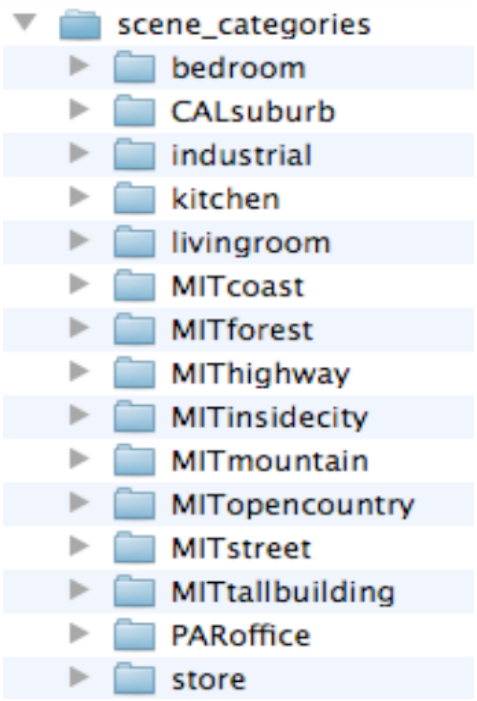


Figura 53. "Ejemplo de ordenación de la base de imágenes"

En las Figuras 54 y 55 se presenta el directorio de los archivos necesarios para el programa. En la Figura se muestra el subdirectorio de GIST y el subdirectorio de BOW.

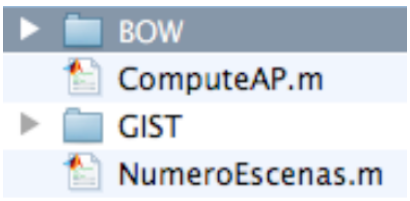


Figura 54. "Directorio del programa"

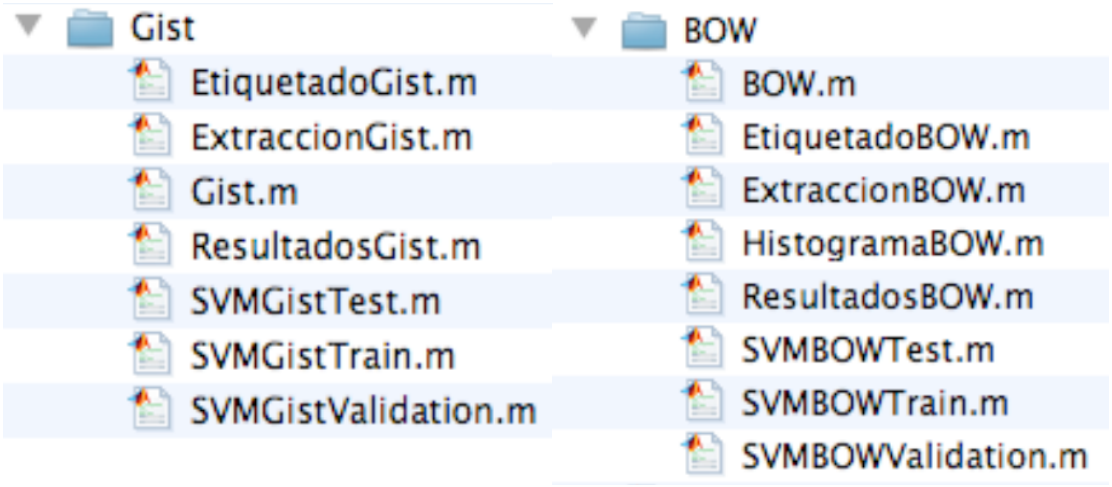


Figura 55. "Subdirectorios del programa"

El programa funciona en base a dos funciones principales distinguidas según el método de extracción que vayamos a usar, estas son Gist.m y BOW.m, ambas reciben como parámetros de entrada la ruta del directorio de la base de datos que antes hemos definido como DIR.

En el caso de Gist.m contiene las llamadas a las funciones de extracción de características *Gist Descriptor*, etiquetado de datos, clasificación (validación, train y test) y por último, la llamada a los resultados.

En el caso de BOW.m contiene las llamadas a las funciones de extracción de características *Bag of words*, normalización de histogramas, etiquetado de datos, clasificación (validación, *train* y *test*) y por último, la llamada a los resultados.

Las funciones auxiliares de este programa que están disponibles para el resto de funciones son:

- NumeroEscenas.m, la cual extrae el nombre de todas las subcarpetas dado un directorio, que después, se utilizarán para escribir cada variable según corresponda a su categoría.

- ComputeAP.m, se usará en los resultados como medida de calidad de los diferentes métodos utilizados. Es la programación del *Average Precision*. Recibe como entradas el vector estimado obtenido en el svmpredict de la fase de *test*, y como segundo parámetro el vector de etiquetas de *test*.

6.3 Archivos programados

6.3.1 GIST

A. Gist.m

```
function Gist(DIR)

% DIR = Directorio de las imágenes

% Extracción Gist
scenes = NumeroEscenas(DIR);
for i = 1:size(scenes,1)
    disp(['Categoría procesada = ' scenes{i} '(' num2str(i) '/'
num2str(size(scenes,1)) ')']);
    if exist([DIR '/' scenes{i}], 'dir') == 7
        ExtraccionGist([DIR '/' scenes{i}], 'DST');
        ExtraccionGist([DIR '/' scenes{i}], 'WDST');
    end
end

% Etiquetado Gist
EtiquetadoGist(DIR, 'DST');
EtiquetadoGist(DIR, 'WDST');
EtiquetadoGist(DIR, 'DST', 'Train');
EtiquetadoGist(DIR, 'WDST', 'Train');
EtiquetadoGist(DIR, 'DST', 'Test');
EtiquetadoGist(DIR, 'WDST', 'Test');

% SVM Validation gist de Train
SVMGistValidation(DIR);

% SVM Train gist de Train
SVMGistTrain(DIR);

% SVM Test gist de Test
SVMGistTest(DIR);

% Resultados de SVM
ResultadosGist(DIR);

end
```

B. ExtraccionGist.m

```
function ExtraccionGist(DIR, transformada)

% Parámetro:
clear param;
param.orientationsPerScale = [8 8 8 8];
if strcmp(transformada, 'DST')
    param.numberBlocks = 1;
elseif strcmp(transformada, 'WDST')
    param.numberBlocks = 4;
end
param.fc_prefilt = 4;

% Computación Gist
[~, scene, ~] = fileparts(DIR);
num = dir(DIR);
contador = 1;
for i = 1:size(num,1)
    [~, ~, ext] = fileparts(num(i).name);
    if strcmp(ext, '.jpg')
        disp(['Imágenes procesadas = '  num2str(contador)]);
        img = imread([DIR '/' num(i).name]);
        eval(['[gist' transformada '_' scene '(contador, :), param] =
LMgist(img, '''', param);']);
        eval(['gist' transformada 'Pattern_' scene '(contador, :) =
{[''' scene '_' num(i).name(1:end-4) ''']};']);
        ExtraccionGistFig(DIR, transformada, num(i).name, eval(['gist'
transformada '_' scene '(contador, :)']), param);
        contador = contador + 1;
    end
end

% Guardar .mat file
eval(['gist' transformada 'Train_' scene ' = gist' transformada '_'
scene '(1:floor(size(gist' transformada '_' scene ',1)*7/10),:);']);
eval(['gist' transformada 'Test_' scene ' = gist' transformada '_' scene
'(floor(size(gist' transformada '_' scene ',1)*7/10)+1:size(gist'
transformada '_' scene ',1),:);']);
save([DIR '/gist' transformada '_' scene], ['gist' transformada '_'
scene], ['gist' transformada 'Train_' scene], ['gist' transformada
'Test_' scene]);
eval(['gist' transformada 'TrainPattern_' scene ' = gist' transformada
'Pattern_' scene '(1:floor(size(gist' transformada 'Pattern_' scene
',1)*7/10),:);']);
eval(['gist' transformada 'TestPattern_' scene ' = gist' transformada
'Pattern_' scene '(floor(size(gist' transformada 'Pattern_' scene
',1)*7/10)+1:size(gist' transformada 'Pattern_' scene ',1),:);']);
save([DIR '/gist' transformada 'Pattern_' scene], ['gist' transformada
'Pattern_' scene], ['gist' transformada 'TrainPattern_' scene], ['gist'
transformada 'TestPattern_' scene]);

end
```

```
function ExtraccionGistFig(DIR, transformada, num, gist, param)

% Crear directorio figures
if exist([DIR '/gist' transformada '_figures'], 'dir') == 0
    mkdir([DIR '/gist' transformada '_figures']);
end

% Visualización y Guardar
fig = figure;
showGist(gist, param);
saveas(fig, [DIR '/gist' transformada '_figures/' num(1:end-4) '.fig'],
'fig');
close(fig);

end
```

C. EtiquetadoGist.m

```
function EtiquetadoGist(DIR, transformada, tipoDatos)

if nargin == 2
    tipoDatos = '';
end

% Cargar gist y gistPattern
scenes = NumeroEscenas(DIR);
for i = 1:size(scenes,1)
    if exist([DIR '/' scenes{i} '/gist' transformada '_' scenes{i}
'.mat'],'file') == 2
        load([DIR '/' scenes{i} '/gist' transformada '_' scenes{i}
'.mat']);
    end
    if exist([DIR '/' scenes{i} '/gist' transformada 'Pattern_'
scenes{i} '.mat'],'file') == 2
        load([DIR '/' scenes{i} '/gist' transformada 'Pattern_'
scenes{i} '.mat']);
    end
end

% Etiquetado de Gist
eval(['gist' transformada tipoDatos ' = [];']);
eval(['gist' transformada tipoDatos 'Pattern = {};']);
for i = 1:size(scenes,1)
    eval(['gist' transformada tipoDatos ' = [gist' transformada
tipoDatos '; gist' transformada tipoDatos '_' scenes{i} '];']);
    eval(['gist' transformada tipoDatos 'Pattern = [gist' transformada
tipoDatos 'Pattern; gist' transformada tipoDatos 'Pattern_' scenes{i}
'];']);
end
matrizRandom = randperm(size(eval(['gist' transformada tipoDatos]),1));
eval(['gist' transformada tipoDatos ' = gist' transformada tipoDatos
'(matrizRandom,:);']);
eval(['gist' transformada tipoDatos 'Pattern = gist' transformada
tipoDatos 'Pattern(matrizRandom,:);']);
for i = 1:size(scenes,1)
    for j = 1:size(eval(['gist' transformada tipoDatos 'Pattern']),1)
        eval(['etiquetado' transformada tipoDatos '_' scenes{i} '(j,:) =
strcmp(scenes{i}, gist' transformada tipoDatos 'Pattern{j,1}(1:end-
11));']);
    end
end

% Guardar Etiquetado
save([DIR '/gist' transformada tipoDatos], ['gist' transformada
tipoDatos], ['gist' transformada tipoDatos 'Pattern'], ['etiquetado'
transformada tipoDatos '_*']);

end
```

D. SVMGistValidation.m

```
function SVMGistValidation(DIR)

% Cargar EtiquetadoTrain
scenes = NumeroEscenas(DIR);
if exist([DIR '/gistDSTTrain.mat'],'file') == 2
    load([DIR '/gistDSTTrain.mat']);
end
if exist([DIR '/gistWDSTTrain.mat'],'file') == 2
    load([DIR '/gistWDSTTrain.mat']);
end

% SVM Validación gist DST de Train
DSTLineal = {'scenes' 'mejorAccLineal' 'mejorCLineal'};
DSTRBF = {'scenes' 'mejorAccRBF' 'mejorCRBF' 'mejorGRBF'};
for i = 1:size(scenes,1)
    % SVM Tipo Lineal
    mejorAccLineal = 0;
    mejorCLineal = 0;cLinealMin = -2;cLinealMax = 2;
    bordeCLinealMin = cLinealMin;bordeCLinealMax = cLinealMax;
    eval(['valoresDSTLineal_' scenes{i} ' = {[''ValidaciÛn de ''
scenes{i} '' DST con SVM Lineal'']};']);
    indice = 1;
    while(1)
        for cLineal = cLinealMin:0.5:cLinealMax
            disp(['DST ' scenes{i} '(' num2str(i) '/'
num2str(size(scenes,1)) ') c=' num2str(cLineal)]);
            positivos = eval(['sum(etiquetadoDSTTrain_' scenes{i}
'==1);']);
            negativos = eval(['sum(etiquetadoDSTTrain_' scenes{i}
'==0);']);
            peso0 = positivos/negativos;peso1 = 1;
            opcionesLineal = ['-s 0 -t 0 -c ' num2str(2^cLineal) ' -v 5
-w0 ' num2str(peso0) ' -w1 ' num2str(peso1)];
            accLineal = svmtrain(double(eval(['etiquetadoDSTTrain_'
scenes{i}])), double(gistDSTTrain), opcionesLineal);
            eval(['valoresDSTLineal_' scenes{i} '(1,indice+1) =
{cLineal};']);
            eval(['valoresDSTLineal_' scenes{i} '(2,indice+1) =
{accLineal};']);
            if (accLineal > mejorAccLineal)
                mejorAccLineal = accLineal;
                mejorCLineal = cLineal;
            end
            indice = indice + 1;
        end
        if (mejorCLineal == bordeCLinealMin)
            bordeCLinealMin = bordeCLinealMin - 2;
            cLinealMax = cLinealMin - 0.5;
            cLinealMin = cLinealMin - 2;
            eval(['valoresDSTLineal_' scenes{i} '(:,2+4:end+4) =
valoresDSTLineal_' scenes{i} '(:,2:end);']);
            eval(['valoresDSTLineal_' scenes{i} '(:,2:5) = {[]};']);
            indice = 1;
        elseif (mejorCLineal == bordeCLinealMax)
            bordeCLinealMax = bordeCLinealMax + 2;
            cLinealMin = cLinealMax + 0.5;
            cLinealMax = cLinealMax + 2;
            indice = eval(['size(valoresDSTLineal_' scenes{i} ',2);']);
        else
            DSTLineal(i+1,:) = {scenes{i} mejorAccLineal mejorCLineal};
```

```
            break;
        end
    end
    % SVM Tipo RBF (Prioridad C)
    mejorAccRBF = 0;
    mejorCRBF = 0;cRBFMin = -2;cRBFMax = 2;
    bordeCRBFMin = cRBFMin;bordeCRBFMax = cRBFMax;
    mejorGRBF = 0;gRBFMin = -2;gRBFMax = 2;
    bordeGRBFMin = gRBFMin;bordeGRBFMax = gRBFMax;
    eval(['valoresDSTRBF_' scenes{i} ' = {[''ValidaciÛn de '' scenes{i}
'' DST con SVM RBF'']};']);
    indiceC = 1;limiteC = 0;indiceG = 1;limiteG = 0;
    while(1)
        for cRBF = cRBFMin:0.5:cRBFMax
            for gRBF = gRBFMin:0.5:gRBFMax
                disp(['DST ' scenes{i} '(' num2str(i) '/'
num2str(size(scenes,1)) ') c=' num2str(cRBF) ' g=' num2str(gRBF)]);
                positivos = eval(['sum(etiquetadoDSTTrain_' scenes{i}
'==1);']);
                negativos = eval(['sum(etiquetadoDSTTrain_' scenes{i}
'==0);']);
                peso0 = positivos/negativos;peso1 = 1;
                opcionesRBF = ['-s 0 -t 2 -c ' num2str(2^cRBF) ' -g '
num2str(2^gRBF) ' -v 5 -w0 ' num2str(peso0) ' -w1 ' num2str(peso1)];
                accRBF = svmtrain(double(eval(['etiquetadoDSTTrain_'
scenes{i}])), double(gistDSTTrain), opcionesRBF);
                eval(['valoresDSTRBF_' scenes{i} '(1,indiceC+1) =
{cRBF};']);
                eval(['valoresDSTRBF_' scenes{i} '(indiceG+1,1) =
{gRBF};']);
                eval(['valoresDSTRBF_' scenes{i} '(indiceG+1,indiceC+1)
= {accRBF};']);
                if (accRBF > mejorAccRBF)
                    mejorAccRBF = accRBF;
                    mejorCRBF = cRBF;
                    mejorGRBF = gRBF;
                end
                indiceG = indiceG + 1;
            end
            indiceG = esLimite(limiteG);
            indiceC = indiceC + 1;
        end
        if (mejorCRBF == bordeCRBFMin)
            bordeCRBFMin = bordeCRBFMin - 2;
            cRBFMax = cRBFMin - 0.5;
            cRBFMin = cRBFMin - 2;
            eval(['valoresDSTRBF_' scenes{i} '(:,2+4:end+4) =
valoresDSTRBF_' scenes{i} '(:,2:end);']);
            eval(['valoresDSTRBF_' scenes{i} '(:,2:5) = {[]};']);
            indiceC = esLimite(limiteC);
            indiceG = esLimite(limiteG);
        elseif (mejorCRBF == bordeCRBFMax)
            bordeCRBFMax = bordeCRBFMax + 2;
            cRBFMin = cRBFMax + 0.5;
            cRBFMax = cRBFMax + 2;
            limiteC = eval(['size(valoresDSTRBF_' scenes{i} ',2);']);
            indiceC = esLimite(limiteC);
            indiceG = esLimite(limiteG);
        elseif (mejorGRBF == bordeGRBFMin)
            bordeGRBFMin = bordeGRBFMin - 2;
            gRBFMax = gRBFMin - 0.5;
            gRBFMin = gRBFMin - 2;
            eval(['valoresDSTRBF_' scenes{i} '(2+4:end+4,:) =
```

```
valoresDSTRBF_' scenes{i} '(2:end,:);']);
            eval(['valoresDSTRBF_' scenes{i} '(2:5,:) = {[]};']);
            indiceC = esLimite(limiteC);
            indiceG = esLimite(limiteG);
        elseif (mejorGRBF == bordeGRBFMax)
            bordeGRBFMax = bordeGRBFMax + 2;
            gRBFMin = gRBFMax + 0.5;
            gRBFMax = gRBFMax + 2;
            limiteG = eval(['size(valoresDSTRBF_' scenes{i} ',1);']);
            indiceC = esLimite(limiteC);
            indiceG = esLimite(limiteG);
        else
            DSTRBF(i+1,:) = {scenes{i} mejorAccRBF mejorCRBF mejorGRBF};
            break;
        end
    end
end

% SVM Validación gist WDST de Train
WDSTLineal = {'scenes' 'mejorAccLineal' 'mejorCLineal'};
WDSTRBF = {'scenes' 'mejorAccRBF' 'mejorCRBF' 'mejorGRBF'};
for i = 1:size(scenes,1)
    % SVM Tipo Lineal
    mejorAccLineal = 0;
    mejorCLineal = 0;cLinealMin = -2;cLinealMax = 2;
    bordeCLinealMin = cLinealMin;bordeCLinealMax = cLinealMax;
    eval(['valoresWDSTLineal_' scenes{i} ' = {[''ValidaciÛn de ''
scenes{i} '' WDST con SVM Lineal'']};']);
    indice = 1;
    while(1)
        for cLineal = cLinealMin:0.5:cLinealMax
            disp(['WDST ' scenes{i} '(' num2str(i) '/'
num2str(size(scenes,1)) ') c=' num2str(cLineal)]);
            positivos = eval(['sum(etiquetadoWDSTTrain_' scenes{i}
'==1);']);
            negativos = eval(['sum(etiquetadoWDSTTrain_' scenes{i}
'==0);']);
            peso0 = positivos/negativos;peso1 = 1;
            opcionesLineal = ['-s 0 -t 0 -c ' num2str(2^cLineal) ' -v 5
-w0 ' num2str(peso0) ' -w1 ' num2str(peso1)];
            accLineal = svmtrain(double(eval(['etiquetadoWDSTTrain_'
scenes{i}])), double(gistWDSTTrain), opcionesLineal);
            eval(['valoresWDSTLineal_' scenes{i} '(1,indice+1) =
{cLineal};']);
            eval(['valoresWDSTLineal_' scenes{i} '(2,indice+1) =
{accLineal};']);
            if (accLineal > mejorAccLineal)
                mejorAccLineal = accLineal;
                mejorCLineal = cLineal;
            end
            indice = indice + 1;
        end
        if (mejorCLineal == bordeCLinealMin)
            bordeCLinealMin = bordeCLinealMin - 2;
            cLinealMax = cLinealMin - 0.5;
            cLinealMin = cLinealMin - 2;
            eval(['valoresWDSTLineal_' scenes{i} '(:,2+4:end+4) =
valoresWDSTLineal_' scenes{i} '(:,2:end);']);
            eval(['valoresWDSTLineal_' scenes{i} '(:,2:5) = {[]};']);
            indice = 1;
        elseif (mejorCLineal == bordeCLinealMax)
            bordeCLinealMax = bordeCLinealMax + 2;
            cLinealMin = cLinealMax + 0.5;
```

```
            cLinealMax = cLinealMax + 2;
            indice = eval(['size(valoresWDSTLineal_' scenes{i} ',2);']);
        else
            WDSTLineal(i+1,:) = {scenes{i} mejorAccLineal mejorCLineal};
            break;
        end
    end
    % SVM Tipo RBF (Prioridad C)
    mejorAccRBF = 0;
    mejorCRBF = 0;cRBFMin = -2;cRBFMax = 2;
    bordeCRBFMin = cRBFMin;bordeCRBFMax = cRBFMax;
    mejorGRBF = 0;gRBFMin = -2;gRBFMax = 2;
    bordeGRBFMin = gRBFMin;bordeGRBFMax = gRBFMax;
    eval(['valoresWDSTRBF_' scenes{i} ' = {[''ValidaciÛn de '' scenes{i}
'' WDST con SVM RBF'']};']);
    indiceC = 1;limiteC = 0;indiceG = 1;limiteG = 0;
    while(1)
        for cRBF = cRBFMin:0.5:cRBFMax
            for gRBF = gRBFMin:0.5:gRBFMax
                disp(['WDST ' scenes{i} '(' num2str(i) '/'
num2str(size(scenes,1)) ') c=' num2str(cRBF) ' g=' num2str(gRBF)]);
                positivos = eval(['sum(etiquetadoWDSTTrain_' scenes{i}
'==1);']);
                negativos = eval(['sum(etiquetadoWDSTTrain_' scenes{i}
'==0);']);
                peso0 = positivos/negativos;peso1 = 1;
                opcionesRBF = ['-s 0 -t 2 -c ' num2str(2^cRBF) ' -g '
num2str(2^gRBF) ' -v 5 -w0 ' num2str(peso0) ' -w1 ' num2str(peso1)];
                accRBF = svmtrain(double(eval(['etiquetadoWDSTTrain_'
scenes{i}])), double(gistWDSTTrain), opcionesRBF);
                eval(['valoresWDSTRBF_' scenes{i} '(1,indiceC+1) =
{cRBF};']);
                eval(['valoresWDSTRBF_' scenes{i} '(indiceG+1,1) =
{gRBF};']);
                eval(['valoresWDSTRBF_' scenes{i} '(indiceG+1,indiceC+1)
= {accRBF};']);
                if (accRBF > mejorAccRBF)
                    mejorAccRBF = accRBF;
                    mejorCRBF = cRBF;
                    mejorGRBF = gRBF;
                end
                indiceG = indiceG + 1;
            end
            indiceG = esLimite(limiteG);
            indiceC = indiceC + 1;
        end
        if (mejorCRBF == bordeCRBFMin)
            bordeCRBFMin = bordeCRBFMin - 2;
            cRBFMax = cRBFMin - 0.5;
            cRBFMin = cRBFMin - 2;
            eval(['valoresWDSTRBF_' scenes{i} '(:,2+4:end+4) =
valoresWDSTRBF_' scenes{i} '(:,2:end);']);
            eval(['valoresWDSTRBF_' scenes{i} '(:,2:5) = {[]};']);
            indiceC = esLimite(limiteC);
            indiceG = esLimite(limiteG);
        elseif (mejorCRBF == bordeCRBFMax)
            bordeCRBFMax = bordeCRBFMax + 2;
            cRBFMin = cRBFMax + 0.5;
            cRBFMax = cRBFMax + 2;
            limiteC = eval(['size(valoresWDSTRBF_' scenes{i} ',2);']);
            indiceC = esLimite(limiteC);
            indiceG = esLimite(limiteG);
        elseif (mejorGRBF == bordeGRBFMin)
```

```
            bordeGRBFMin = bordeGRBFMin - 2;
            gRBFMax = gRBFMin - 0.5;
            gRBFMin = gRBFMin - 2;
            eval(['valoresWDSTRBF_' scenes{i} '(2+4:end+4,:) =
valoresWDSTRBF_' scenes{i} '(2:end,:);']);
            eval(['valoresWDSTRBF_' scenes{i} '(2:5,:) = {[]};']);
            indiceC = esLimite(limiteC);
            indiceG = esLimite(limiteG);
        elseif (mejorGRBF == bordeGRBFMax)
            bordeGRBFMax = bordeGRBFMax + 2;
            gRBFMin = gRBFMax + 0.5;
            gRBFMax = gRBFMax + 2;
            limiteG = eval(['size(valoresWDSTRBF_' scenes{i} ',1);']);
            indiceC = esLimite(limiteC);
            indiceG = esLimite(limiteG);
        else
            WDSTRBF(i+1,:) = {scenes{i} mejorAccRBF mejorCRBF
mejorGRBF};
            break;
        end
    end
end

% Guardar SVM Validación
save([DIR '/SVMgistTrain'], 'DSTLineal', 'DSTRBF', 'WDSTLineal',
'WDSTRBF', 'valoresDSTLineal_*', 'valoresDSTRBF_*',
'valoresWDSTLineal_*', 'valoresWDSTRBF_*');

end


function indice = esLimite(limite)

% Comprobación del límite
if(limite ~= 0)
    indice = limite;
else
    indice = 1;
end

end
```

E. SVMGistTrain.m

```
function SVMGistTrain(DIR)

% Cargar EtiquetadoTrain
scenes = NumeroEscenas(DIR);
if exist([DIR '/gistDSTTrain.mat'],'file') == 2
    load([DIR '/gistDSTTrain.mat']);
end
if exist([DIR '/gistWDSTTrain.mat'],'file') == 2
    load([DIR '/gistWDSTTrain.mat']);
end

% Cargar SVM Validación
if exist([DIR '/SVMgistTrain.mat'],'file') == 2
    load([DIR '/SVMgistTrain.mat']);
end

% SVM Train gist DST de Train
for i = 1:size(scenes,1)
    positivosPOS = eval(['find(etiquetadoDSTTrain_' scenes{i} '==1);']);
    negativosPOS = eval(['find(etiquetadoDSTTrain_' scenes{i} '==0);']);
    etiquetasTEMP = eval(['etiquetadoDSTTrain_' scenes{i}
'([positivosPOS;negativosPOS],:);']);
    gistTEMP = gistDSTTrain([positivosPOS;negativosPOS],:);
    positivos = eval(['sum(etiquetadoDSTTrain_' scenes{i} '==1);']);
    negativos = eval(['sum(etiquetadoDSTTrain_' scenes{i} '==0);']);
    peso0 = positivos/negativos;peso1 = 1;
    opcionesLineal = ['-s 0 -t 0 -c ' num2str(2^DSTLineal{i+1,3}) ' -w0
' num2str(peso0) ' -w1 ' num2str(peso1)];
    eval(['ModeloDSTLineal_' scenes{i} ' =
svmtrain(double(etiquetasTEMP), double(gistTEMP), opcionesLineal);']);
    opcionesRBF = ['-s 0 -t 2 -c ' num2str(2^DSTRBF{i+1,3}) ' -g '
num2str(2^DSTRBF{i+1,4}) ' -w0 ' num2str(peso0) ' -w1 ' num2str(peso1)];
    eval(['ModeloDSTRBF_' scenes{i} ' = svmtrain(double(etiquetasTEMP),
double(gistTEMP), opcionesRBF);']);
end

% SVM Train gist WDST de Train
for i = 1:size(scenes,1)
    positivosPOS = eval(['find(etiquetadoWDSTTrain_' scenes{i}
'==1);']);
    negativosPOS = eval(['find(etiquetadoWDSTTrain_' scenes{i}
'==0);']);
    etiquetasTEMP = eval(['etiquetadoWDSTTrain_' scenes{i}
'([positivosPOS;negativosPOS],:);']);
    gistTEMP = gistWDSTTrain([positivosPOS;negativosPOS],:);
    positivos = eval(['sum(etiquetadoWDSTTrain_' scenes{i} '==1);']);
    negativos = eval(['sum(etiquetadoWDSTTrain_' scenes{i} '==0);']);
    peso0 = positivos/negativos;peso1 = 1;
    opcionesLineal = ['-s 0 -t 0 -c ' num2str(2^WDSTLineal{i+1,3}) ' -w0
' num2str(peso0) ' -w1 ' num2str(peso1)];
    eval(['ModeloWDSTLineal_' scenes{i} ' =
svmtrain(double(etiquetasTEMP), double(gistTEMP), opcionesLineal);']);
    opcionesRBF = ['-s 0 -t 2 -c ' num2str(2^WDSTRBF{i+1,3}) ' -g '
num2str(2^WDSTRBF{i+1,4}) ' -w0 ' num2str(peso0) ' -w1 '
num2str(peso1)];
    eval(['ModeloWDSTRBF_' scenes{i} ' = svmtrain(double(etiquetasTEMP),
double(gistTEMP), opcionesRBF);']);
end
```

```
% Guardar SVMModelos
save([DIR '/SVMgistModelos'], 'ModeloDSTLineal_*', 'ModeloDSTRBF_*',
'ModeloWDSTLineal_*', 'ModeloWDSTRBF_*');

end
```

F. SVMGistTest.m

```
function SVMGistTest(DIR)

% Cargar EtiquetadoTest
scenes = NumeroEscenas(DIR);
if exist([DIR '/gistDSTTest.mat'],'file') == 2
    load([DIR '/gistDSTTest.mat']);
end
if exist([DIR '/gistWDSTTest.mat'],'file') == 2
    load([DIR '/gistWDSTTest.mat']);
end

% Cargar SVMModelos
if exist([DIR '/SVMgistModelos.mat'],'file') == 2
    load([DIR '/SVMgistModelos.mat']);
end

% SVM Test gist DST de Test
for i = 1:size(scenes,1)
    eval(['[PronosticoDSTLineal_' scenes{i} ', PrecisionDSTLineal_'
scenes{i} ', EstimadoDSTLineal_' scenes{i} '] =
svmpredict(double(etiquetadoDSTTest_' scenes{i} '), double(gistDSTTest),
ModeloDSTLineal_' scenes{i} ', ''''');']);
    eval(['[PronosticoDSTRBF_' scenes{i} ', PrecisionDSTRBF_' scenes{i}
', EstimadoDSTRBF_' scenes{i} '] = svmpredict(double(etiquetadoDSTTest_'
scenes{i} '), double(gistDSTTest), ModeloDSTRBF_' scenes{i} ',
''''');']);
end

% SVM Test gist WDST de Test
for i = 1:size(scenes,1)
    eval(['[PronosticoWDSTLineal_' scenes{i} ', PrecisionWDSTLineal_'
scenes{i} ', EstimadoWDSTLineal_' scenes{i} '] =
svmpredict(double(etiquetadoWDSTTest_' scenes{i} '),
double(gistWDSTTest), ModeloWDSTLineal_' scenes{i} ', ''''');']);
    eval(['[PronosticoWDSTRBF_' scenes{i} ', PrecisionWDSTRBF_'
scenes{i} ', EstimadoWDSTRBF_' scenes{i} '] =
svmpredict(double(etiquetadoWDSTTest_' scenes{i} '),
double(gistWDSTTest), ModeloWDSTRBF_' scenes{i} ', ''''');']);
end

% Guardar SVM Test
save([DIR '/SVMgistPronosticos'], 'PronosticoDSTLineal_*',
'PronosticoDSTRBF_*', 'PronosticoWDSTLineal_*', 'PronosticoWDSTRBF_*');
save([DIR '/SVMgistPrecision'], 'PrecisionDSTLineal_*',
'PrecisionDSTRBF_*', 'PrecisionWDSTLineal_*', 'PrecisionWDSTRBF_*');
save([DIR '/SVMgistEstimado'], 'EstimadoDSTLineal_*',
'EstimadoDSTRBF_*', 'EstimadoWDSTLineal_*', 'EstimadoWDSTRBF_*');

end
```

G. ResultadosGist.m

```
function resultados = ResultadosGist(DIR)

% Cargar EtiquetadoTest
scenes = NumeroEscenas(DIR);
if exist([DIR '/gistDSTTest.mat'],'file') == 2
    load([DIR '/gistDSTTest.mat']);
end
if exist([DIR '/gistWDSTTest.mat'],'file') == 2
    load([DIR '/gistWDSTTest.mat']);
end

% Cargar SVM Test Estimado
if exist([DIR '/SVMgistEstimado.mat'],'file') == 2
    load([DIR '/SVMgistEstimado.mat']);
end

% Matriz de resultados
resultados(:,1) = {'Resultados' 'DST Lineal' 'DST RBF' 'WDST Lineal'
'WDST RBF'};
for i = 1:size(scenes,1)
    resultados(1,i+1) = {scenes(i)};
    resultados(2,i+1) = eval(['{ComputeAP(EstimadoDSTLineal_' scenes{i}
',etiquetadoDSTTest_' scenes{i} ')*100};']);
    resultados(3,i+1) = eval(['{ComputeAP(EstimadoDSTRBF_' scenes{i}
',etiquetadoDSTTest_' scenes{i} ')*100};']);
    resultados(4,i+1) = eval(['{ComputeAP(EstimadoWDSTLineal_' scenes{i}
',etiquetadoWDSTTest_' scenes{i} ')*100};']);
    resultados(5,i+1) = eval(['{ComputeAP(EstimadoWDSTRBF_' scenes{i}
',etiquetadoWDSTTest_' scenes{i} ')*100};']);
end
resultados(1,i+2) = {'Average'};
resultados(2,i+2) = {mean(cell2mat(resultados(2,2:end-1)))};
resultados(3,i+2) = {mean(cell2mat(resultados(3,2:end-1)))};
resultados(4,i+2) = {mean(cell2mat(resultados(4,2:end-1)))};
resultados(5,i+2) = {mean(cell2mat(resultados(5,2:end-1)))};

% Guardar SVMResultados
save([DIR '/SVMgistResultados'], 'resultados');

end
```

6.3.2 BOW

A. BOW.m

```
function BOW(DIR)

% DIR = Directorio de las imágenes

% Extracción BOW
scenes = NumeroEscenas(DIR);
for i = 1:size(scenes,1)
    disp(['Categoría procesada = ' scenes{i} '(' num2str(i) '/'
num2str(size(scenes,1)) ')']);
    if exist([DIR '/' scenes{i}], 'dir') == 7
        ExtraccionBOW([DIR '/' scenes{i}]);
    end
end

% Histogramas BOW
muestras = 100000;
centroides = 1000;
HistogramaBOW(DIR, muestras, centroides);

% Etiquetado BOW
EtiquetadoBOW(DIR);
EtiquetadoBOW(DIR, 'Train');
EtiquetadoBOW(DIR, 'Test');

% SVM Validation BOW de Train
SVMBOWValidation(DIR);

% SVM Train BOW de Train
SVMBOWTrain(DIR);

% SVM Test BOW de Test
SVMBOWTest(DIR);

% Resultados de SVM
ResultadosBOW(DIR);

end
```

B. ExtraccionBOW.m

```
function ExtraccionBOW(DIR)

% Computación BOW
[~, scene, ~] = fileparts(DIR);
num = dir(DIR);
eval(['numimgs_' scene ' = 0;']);
for i = 1:size(num,1)
    [~, ~, ext] = fileparts(num(i).name);
    if strcmp(ext, '.jpg')
        disp(['Imágenes procesadas = '  num2str(eval(['numimgs_' scene '
+ 1']))]);
        if size(imread([DIR '/' num(i).name]),3) == 3
            img = single(rgb2gray(imread([DIR '/' num(i).name])));
        else
            img = single(imread([DIR '/' num(i).name]));
        end
        eval(['[feat, BOW_' scene '_' num(i).name(1:end-4) '] =
vl_sift(img);']);
        ExtraccionBOWFig(DIR, num(i).name, img, feat);
        eval(['numimgs_' scene ' = numimgs_' scene ' + 1;']);
    end
end

% Guardar .mat file
save([DIR '/BOW_' scene], ['BOW_' scene '_*'], ['numimgs_' scene]);

end

function ExtraccionBOWFig(DIR, num, img, feat)

% Crear directorio figures
if exist([DIR '/BOW_figures'], 'dir') == 0
    mkdir([DIR '/BOW_figures']);
end

% Visualización y Guardar
fig = figure;
image(img);
vl_plotframe(feat);
saveas(fig, [DIR '/BOW_figures/' num(1:end-4) '.fig'], 'fig');
close(fig);

end
```

C. HistogramaBOW.m

```
function HistogramaBOW(DIR, muestras, centroides)

% Cargar BOW
scenes = NumeroEscenas(DIR);
contador = 0;
for i = 1:size(scenes,1)
    if exist([DIR '/' scenes{i} '/BOW_' scenes{i} '.mat'],'file') == 2
        load([DIR '/' scenes{i} '/BOW_' scenes{i} '.mat']);
    end
    contador = eval(['contador + numimgs_' scenes{i}]);
end

% Computación BOWHistograma
BOWset = [];
for i = 1:size(scenes,1)
    num = dir([DIR '/' scenes{i}]);
    for j = 1:size(num,1)
        [~, ~, ext] = fileparts(num(j).name);
        if strcmp(ext, '.jpg')
            matrizRandom = randperm(size(eval(['BOW_' scenes{i} '_'
num(j).name(1:end-4)]),2), round(muestras/contador));
            BOWset = eval(['[BOWset, BOW_' scenes{i} '_'
num(j).name(1:end-4) '(:,matrizRandom)];']);
        end
    end
end
BOWcentroides = vl_ikmeans(BOWset, centroides, 'method', 'elkan');
for i = 1:size(scenes,1)
    contador = 1;
    num = dir([DIR '/' scenes{i}]);
    for j = 1:size(num,1)
        [~, ~, ext] = fileparts(num(j).name);
        if strcmp(ext, '.jpg')
            BOWindeces = vl_ikmeanspush(eval(['BOW_' scenes{i} '_'
num(j).name(1:end-4)]), BOWcentroides);
            BOWhistograma = vl_ikmeanshist(centroides, BOWindeces);
            eval(['BOW_' scenes{i} '(:,contador) =
BOWhistograma/sum(BOWhistograma);']);
            eval(['BOWPattern_' scenes{i} '(:,contador) = {['''
scenes{i} '_' num(j).name(1:end-4) '''']};']);
            contador = contador + 1;
        end
    end
end

% Guardar BOWHistograma
for i = 1:size(scenes,1)
    save([DIR '/BOWCentroides'], 'BOWcentroides');
    eval(['BOWTrain_' scenes{i} ' = BOW_' scenes{i}
'(:,1:floor(size(BOW_' scenes{i} ',2)*7/10));']);
    eval(['BOWTest_' scenes{i} ' = BOW_' scenes{i} '(:,floor(size(BOW_'
scenes{i} ',2)*7/10)+1:size(BOW_' scenes{i} ',2));']);
    save([DIR '/' scenes{i} '/BOWNormalizado_' scenes{i}], ['BOW_'
scenes{i}], ['BOWTrain_' scenes{i}], ['BOWTest_' scenes{i}]);
    eval(['BOWTrainPattern_' scenes{i} ' = BOWPattern_' scenes{i}
'(:,1:floor(size(BOWPattern_' scenes{i} ',2)*7/10));']);
    eval(['BOWTestPattern_' scenes{i} ' = BOWPattern_' scenes{i}
'(:,floor(size(BOWPattern_' scenes{i} ',2)*7/10)+1:size(BOWPattern_'
scenes{i} ',2));']);
    save([DIR '/' scenes{i} '/BOWPattern_' scenes{i}], ['BOWPattern_'
```

```
scenes{i}], ['BOWTrainPattern_' scenes{i}], ['BOWTestPattern_'
scenes{i}]);
end

end
```

D. EtiquetadoBOW.m

```
function EtiquetadoBOW(DIR, tipoDatos)

if nargin == 1
    tipoDatos = '';
end

% Cargar BOW
scenes = NumeroEscenas(DIR);
for i = 1:size(scenes,1)
    if exist([DIR '/' scenes{i} '/BOWNormalizado_' scenes{i}
'.mat'],'file') == 2
        load([DIR '/' scenes{i} '/BOWNormalizado_' scenes{i} '.mat']);
    end
    if exist([DIR '/' scenes{i} '/BOWPattern_' scenes{i} '.mat'],'file')
== 2
        load([DIR '/' scenes{i} '/BOWPattern_' scenes{i} '.mat']);
    end
end

% Etiquetado de BOW
eval(['BOW' tipoDatos ' = [];']);
eval(['BOW' tipoDatos 'Pattern = {};']);
for i = 1:size(scenes,1)
    eval(['BOW' tipoDatos ' = [BOW' tipoDatos ' BOW' tipoDatos '_'
scenes{i} '];']);
    eval(['BOW' tipoDatos 'Pattern = [BOW' tipoDatos 'Pattern BOW'
tipoDatos 'Pattern_' scenes{i} '];']);
end
matrizRandom = randperm(size(eval(['BOW' tipoDatos]),2));
eval(['BOW' tipoDatos ' = BOW' tipoDatos '(:,matrizRandom);']);
eval(['BOW' tipoDatos 'Pattern = BOW' tipoDatos
'Pattern(:,matrizRandom);']);
for i = 1:size(scenes,1)
    for j = 1:size(eval(['BOW' tipoDatos 'Pattern']),2)
        eval(['etiquetado' tipoDatos '_' scenes{i} '(:,j) =
strcmp(scenes{i}, BOW' tipoDatos 'Pattern{1,j}(1:end-11));']);
    end
end

% Guardar Etiquetado
save([DIR '/BOW' tipoDatos], ['BOW' tipoDatos], ['BOW' tipoDatos
'Pattern'], ['etiquetado' tipoDatos '_*']);

end
```

E. SVMBOWValidation.m

```
function SVMBOWValidation(DIR)

% Cargar EtiquetadoTrain
scenes = NumeroEscenas(DIR);
if exist([DIR '/BOWTrain.mat'],'file') == 2
    load([DIR '/BOWTrain.mat']);
end

% SVM Validación BOW de Train
kernelTrain = kernel_matrix(BOWTrain',1);
kernelTrain = [(1:size(BOWTrain, 2))' kernelTrain];
BOWKernelExterno = {'scenes' 'mejorAccKernelExterno'
'mejorCKernelExterno'};
for i = 1:size(scenes,1)
    % SVM con kernel externo
    mejorAccKernelExterno = 0;
    mejorCKernelExterno = 0;cKernelExternoMin = -2;cKernelExternoMax =
2;
    bordeCKernelExternoMin = cKernelExternoMin;bordeCKernelExternoMax =
cKernelExternoMax;
    eval(['valoresSVMBOW_' scenes{i} ' = {[''ValidaciÛn de '' scenes{i}
'' BOW con SVM Kernel Externo'']};']);
    indice = 1;
    while(1)
        for cKernelExterno = cKernelExternoMin:0.5:cKernelExternoMax
            disp(['BOW ' scenes{i} '(' num2str(i) '/'
num2str(size(scenes,1)) ') c=' num2str(cKernelExterno)]);
            positivos = eval(['sum(etiquetadoTrain_' scenes{i}
'==1);']);
            negativos = eval(['sum(etiquetadoTrain_' scenes{i}
'==0);']);
            peso0 = positivos/negativos;peso1 = 1;
            opcionesKernelExterno = ['-s 0 -t 4 -c '
num2str(2^cKernelExterno) ' -v 5 -w0 ' num2str(peso0) ' -w1 '
num2str(peso1)];
            accKernelExterno = svmtrain(double(eval(['etiquetadoTrain_'
scenes{i} ''''])), kernelTrain, opcionesKernelExterno);
            eval(['valoresSVMBOW_' scenes{i} '(1,indice+1) =
{cKernelExterno};']);
            eval(['valoresSVMBOW_' scenes{i} '(2,indice+1) =
{accKernelExterno};']);
            if (accKernelExterno > mejorAccKernelExterno)
                mejorAccKernelExterno = accKernelExterno;
                mejorCKernelExterno = cKernelExterno;
            end
            indice = indice + 1;
        end
        if (mejorCKernelExterno == bordeCKernelExternoMin)
            bordeCKernelExternoMin = bordeCKernelExternoMin - 2;
            cKernelExternoMax = cKernelExternoMin - 0.5;
            cKernelExternoMin = cKernelExternoMin - 2;
            eval(['valoresSVMBOW_' scenes{i} '(:,2+4:end+4) =
valoresSVMBOW_' scenes{i} '(:,2:end);']);
            eval(['valoresSVMBOW_' scenes{i} '(:,2:5) = {[]};']);
            indice = 1;
        elseif (mejorCKernelExterno == bordeCKernelExternoMax)
            bordeCKernelExternoMax = bordeCKernelExternoMax + 2;
            cKernelExternoMin = cKernelExternoMax + 0.5;
            cKernelExternoMax = cKernelExternoMax + 2;
            indice = eval(['size(valoresSVMBOW_' scenes{i} ',2);']);
```

```
        else
            BOWKernelExterno(i+1,:) = {scenes{i} mejorAccKernelExterno
mejorCKernelExterno};
            break;
        end
    end
end

% Guardar SVM Validación
save([DIR '/SVMBOWTrain'], 'BOWKernelExterno', 'valoresSVMBOW_*');

end
```

F. SVMBOWTrain.m

```
function SVMBOWTrain(DIR)

% Cargar EtiquetadoTrain
scenes = NumeroEscenas(DIR);
if exist([DIR '/BOWTrain.mat'],'file') == 2
    load([DIR '/BOWTrain.mat']);
end

% Cargar SVM Validación
if exist([DIR '/SVMBOWTrain.mat'],'file') == 2
    load([DIR '/SVMBOWTrain.mat']);
end

% SVM Train BOW de Train
for i = 1:size(scenes,1)
    positivosPOS = eval(['find(etiquetadoTrain_' scenes{i} '==1);']);
    negativosPOS = eval(['find(etiquetadoTrain_' scenes{i} '==0);']);
    etiquetasTEMP = eval(['etiquetadoTrain_' scenes{i} '([positivosPOS
negativosPOS]);']);
    BOWTEMP = BOWTrain(:,[positivosPOS negativosPOS]);
    kernelTEMP = kernel_matrix(BOWTEMP', 1);
    kernelTEMP = [(1:size(BOWTEMP, 2))' kernelTEMP];
    positivos = eval(['sum(etiquetadoTrain_' scenes{i} '==1);']);
    negativos = eval(['sum(etiquetadoTrain_' scenes{i} '==0);']);
    peso0 = positivos/negativos;peso1 = 1;
    opcionesKernelExterno = ['-s 0 -t 4 -c '
num2str(2^BOWKernelExterno{i+1,3}) ' -w0 ' num2str(peso0) ' -w1 '
num2str(peso1)];
    eval(['ModeloSVMBOW_' scenes{i} ' =
svmtrain(double(etiquetasTEMP''), kernelTEMP,
opcionesKernelExterno);']);
end

% Guardar SVMModelos
save([DIR '/SVMBOWModelos'], 'ModeloSVMBOW_*');

end
```

G. SVMBOWTest.m

```
function SVMBOWTest(DIR)

% Cargar EtiquetadoTrain
scenes = NumeroEscenas(DIR);
if exist([DIR '/BOWTrain.mat'],'file') == 2
    load([DIR '/BOWTrain.mat']);
end

% Cargar EtiquetadoTest
if exist([DIR '/BOWTest.mat'],'file') == 2
    load([DIR '/BOWTest.mat']);
end

% Cargar SVMModelos
if exist([DIR '/SVMBOWModelos.mat'],'file') == 2
    load([DIR '/SVMBOWModelos.mat']);
end

% SVM Test BOW de Test
for i = 1:size(scenes,1)
    positivosPOS = eval(['find(etiquetadoTrain_' scenes{i} '==1);']);
    negativosPOS = eval(['find(etiquetadoTrain_' scenes{i} '==0);']);
    etiquetasTEMP = eval(['etiquetadoTrain_' scenes{i} '([positivosPOS
negativosPOS]);']);
    BOWTEMP = BOWTrain(:,[positivosPOS negativosPOS]);
    kernelTest = kernel_matrix(BOWTest', 1, BOWTEMP');
    kernelTest = [(1:size(BOWTest, 2))' kernelTest];
    eval(['[PronosticoSVMBOW_' scenes{i} ', PrecisionSVMBOW_' scenes{i}
', EstimadoSVMBOW_' scenes{i} '] = svmpredict(double(etiquetadoTest_'
scenes{i} '''), kernelTest, ModeloSVMBOW_' scenes{i} ', '''');']);
end

% Guardar SVM Test
save([DIR '/SVMBOWPronosticos'], 'PronosticoSVMBOW_*');
save([DIR '/SVMBOWPrecision'], 'PrecisionSVMBOW_*');
save([DIR '/SVMBOWEstimado'], 'EstimadoSVMBOW_*');

end
```

H. ResultadosBOW.m

```
function resultados = ResultadosBOW(DIR)

% Cargar EtiquetadoTest
scenes = NumeroEscenas(DIR);
if exist([DIR '/BOWTest.mat'],'file') == 2
    load([DIR '/BOWTest.mat']);
end

% Cargar SVM Test Estimado
if exist([DIR '/SVMBOWEstimado.mat'],'file') == 2
    load([DIR '/SVMBOWEstimado.mat']);
end

% Matriz de resultados
resultados(:,1) = {'Resultados' 'Kernel Externo'};
for i = 1:size(scenes,1)
    resultados(1,i+1) = {scenes(i)};
    resultados(2,i+1) = eval(['{ComputeAP(EstimadoSVMBOW_' scenes{i}
',etiquetadoTest_' scenes{i} ')*100};']);
end
resultados(1,i+2) = {'Average'};
resultados(2,i+2) = {mean(cell2mat(resultados(2,2:end-1)))};

% Guardar SVMResultados
save([DIR '/SVMBOWResultados'], 'resultados');

end
```

6.3.3 Funciones auxiliares

A. NumeroEscenas.m

```
function scenes = NumeroEscenas(DIR)

if nargin == 0
    DIR = pwd;
end

scenes = dir(DIR);
folder = scenes(4:end);
scenes = {};
cont = 1;
for i = 1:size(folder,1)
    if exist([DIR '/' folder(i).name], 'dir') == 7
        scenes{cont,1} = folder(i).name;
        cont = cont + 1;
    end
end

end
```

B. ComputeAP.m

```
function ap = ComputeAP(outs,gtLabels)

[~,si] = sort(-outs);
tp = gtLabels(si)>0;
tp = cumsum(tp);
fp = gtLabels(si)<=0;
fp = cumsum(fp);
rec = tp/sum(gtLabels>0);
prec = tp./(fp+tp);

% Compute Average Precision
ap = 0;
for t = 0:0.1:1
    p = max(prec(rec>=t));
    if isempty(p)
        p = 0;
    end
    ap = ap+p/11;
end

end
```

Glosario

.zip	*Formato de almacenamiento sin perdidas, utilizado para la compresión de datos*
Accuracy	*Medida de calidad que determina la exactitud del clasificador*
AP	*Average Precisión*
BoW	*Bag of words, método de extracción de características utilizado en el proyecto*
CBIR	*Content Based Image Retrieval*
Dense SIFT	*Tipo de detector SIFT que crea una malla en toda la imagen para extraer sus descriptores.*
DFT	*Discrete Fourier Transform*
DoG	*Difference of Gaussians*
DST	*Discriminant Spectral Templates*
Gist descriptor	*Descriptores de esencia de la imagen, método de extracción de características utilizado en el proyecto*

IDE	*Integrated Development Environment*
K-means	*Algoritmo usado para el agrupamiento de datos*
Kernel	*Traducido como nucleo, se utiliza en algoritmos de aprendizaje automático*
Keypoints	*Puntos de interés para una imagen que extrae la transformada SIFT*
KLT	*Karhunen-Loeve Transform*
MSER	*Maximally Stable Extremal Regions*
Outlier	*En estadística, es un dato que se encuentra alejado del resto de los datos.*
Path	*Significa ruta. Es la forma de referenciar un archive o directorio en un sistema operativo*
Precision	*Medida de calidad que determina la precisión*
RBF	*Radial Basis Function*
Recall	*Medida de calidad que determina el recuerdo*
SIFT	*Scale Invariant Feature Transform*
Sparse	*Carácter que puede tener un histograma escaso, esparcido y poco denso.*
Spatial Envelope	*Envoltura espacial, concepto para definir Gist descriptor*
Spatial Pyramid Matching	*Algoritmo que emplea una función kernel para crear un mapa de histogramas Bag of words en toda la imagen*
SVM	*Support Vector Machine, técnica de clasificación*
Toolbox	*Literalmente significa caja de herramientas. En informática el uso de esta palabra se refiere a herramientas o librerías que contienen programas de una funcionalidad concreta*
WDST	*Windowed Discriminant Spectral Templates*
WFT	*Windowed Fourier Transform*

Referencias

[1] **Matlab Help.** "The Mathworks", 2005.

[2] **Aude Oliva, Antonio Torralba.** "Modeling the shape of the scene: a holistic representation of the spatial envelope", *International Journal of Computer Vision*, Vol. 42(3): 145-175, 2001.

[3] **Gabriella Csurka, Christopher R.Dance, Lixin Fan, Jutta Willamowski.** "Visual Categorization with Bags of Keypoints", *Xerox Research Centre Europe*, 2004.

[4] **Iván González Díaz.** "Generative Models for image segmentation and representation", Tesis doctoral, Universidad Carlos III de Madrid, 2011.

[5] **Svetlana Lazebnik, Cordelia Schmid, Jean Ponce.** "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* New York, June 2006, vol. II, pp. 2169-2178.

[6] **Sara Pardo Feijoo.** "Aplicación del modelo Bag-of-words al reconocimiento de imágenes", Proyecto fin de carrera, Universidad Carlos III de Madrid, 2009.

[7] **VLFeat 0.9.16.** http://www.vlfeat.org

[8] **LibSVM 3.12.** http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[9] **Datasets for Computer Vision Research. Fifteen Scene Categories.** http://www-cvr.ai.uiuc.edu/ponce_grp/data/

[10] **Yasuhide Mori, Hironobu Takahashi and Ryuichi Oka.** "Image-to-word transformation base don dividing and vector quantizing images with words", *First International Workshop on Multimedia Intelligent Storage and Retrieval Manegement* (MISRM'99), December 1999. http://www.info.uqam.ca/~misrm/papers/mori.ps

[11] **LabelMe**. http://labelme.csail.mit.edu/Release3.0/

[12] **Estes, J. E. Estes, J. E., Hajic, E. J. and L. R. Tinney.** "Fundamentals of Image Analysis: Analysis of Visible and Thermal Infrared Data, Manual of Remote Sensing, American Society for Photogrammetry & Remote Sensing", *Analysis of Visible and Thermal Infrared Data, Manual of Remote Sensing*, American Society for Photogrammetry & Remote Sensing, Bethesda, pp. 987-1124, 1983.

[13] **Jensen J. R..** "Remote Sensing of The Environment: An Earth Resource Perspective", 2nd Edition, *Upper Saddle River: Prentice-Hall*, 592 p, 2000.

[14] **Wikipedia.** "Reconocimiento de patrones" http://es.wikipedia.org/wiki/Reconocimiento_de_patrones

[15] **Shigeo Abe.** "Pattern classification. Neuro-fuzzy methods and their comparison", London ; New York : *Springer (Book),* 2001.

[16] **Wikipedia.** "Accuracy paradox" http://en.wikipedia.org/wiki/Accuracy_paradox

[17] **Mu Zhu.** "Recall, Precision and Average Precision", *working paper* 2004-09 department of statistics & actuarial science university of waterloo august 26, 2004.

[18] **Wikipedia.** "Information retrieval, Average Precision" http://en.wikipedia.org/wiki/Information_retrieval#Average_precision

[19] **DTRG Software For Predictive Modeling and Forecasting.** "Introduction to Support Vector Machines (SVM) Models". http://www.dtreg.com/svm.htm

[20] **Andrea Vedaldi and Andrew Zisserman.** "Image Classification Practical", *Département d'informatique de l'ENS*, 2011.

[21] **Chih-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin.** "A Practical Guide to Support Vector Classification", *Department of Computer Science.National Taiwan University*, Taipei 106,Taiwan, 2010. http://www.csie.ntu.edu.tw/~cjlin

[22] **C. Cortes and V. Vapnik.** "Support-vector network". *Machine Learning* (Book), 20, 1–25, 1995.

[23] **W. S. Sarle.** "Neural Network", Periodic posting to the Usenet newsgroup comp.ai.neural-nets, 1997. ftp://ftp.sas.com/pub/neural/FAQ.html

[24] **Wikipedia.** "Support Vector Machine" http://en.wikipedia.org/wiki/Support_vector_machine

[25] **Biederman, I.** "Aspects and extensión of a theory of human image understanding. In Computational Processes in Human Vision: An Interdisciplinary Perspective", Z. Pylyshyn (Ed.), *Ablex Publishing Corporation: Norwood*, New Jersey, 1988.

[26] **Wikipedia.** "Machine learning" http://en.wikipedia.org/wiki/Machine_learning

[27] **Sanocki, T. And Epstein, W.** "Priming spatial layout of scenes", *Psychological Science,* 8, 374-378, 1997.

[28] **Ripley, B. D.** "Distinctive image features from scale-invariant keypoints", *Cambridge University Press*, 1996.

[29] **Lowe, D. G.** "Pattern Recognition and Neural Networks", *International Journal of Computer Vision*. 60, 2. Pp. 91-110, 2004.

[30] **Gist Descriptor.**
http://people.csail.mit.edu/torralba/code/spatialenvelope/gistdescriptor.zip

[31] **Wikipedia.** "Gabor filter" http://en.wikipedia.org/wiki/Gabor_filter

[32] **J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle. "**Least Squares Support Vector Machines", *World Scientific Pub. Co.*, Singapore, 2002 (ISBN 981-238-151-1). http://www.esat.kuleuven.be/sista/lssvmlab/

[33] **K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, J.A.K. Suykens.** "LS-SVMlab Toolbox User's Guide", *Internal Report 10-146*, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2010.

[34] **Jia Li and James Z.Wang.** "Real-Time Computerized Annotation of Pictures", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):985-1002, 2008.

[35] **Masashi Inoue.** "On the need for annotation-based image retrieval", *CM Computing Surveys* 40 (2): 1–60, 2008.

[36] **A. Grace Selvarani y Dr. S. Annadurai.** "Medical image retrieval by combining low level features and Dicom features", *IEEE International conference on computational intelligence and multimedia applications (ICCIMA)*, vol. 1, pp.587- 589, 2007.

[37] **Li-Jia Li, Richard Socher and Li Fei-Fei.** "Towards Total Scene Understanding: Classification, Annotation and Segmentation in an Automatic Framework", *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[38] **Dong Nia, Yim Pan Chuia, Yingge Qua, Xuan Yanga, Jing Qina, Tien-Tsin Wonga, Simon S.H. Hob, Pheng Ann Henga.** "Reconstruction of volumetric ultrasound panorama based on improved 3D SIFT", *Computerized Medical Imaging and Graphics, Volume 33*, Issue 7, Pages 559-566, ISSN 0895-6111, October 2009.

[39] **Instructor: James Hays.** "Introduction to Computer Vision", *CS143 of Brown*

Computer Science. Columbia School. (Course) http://www.cs.brown.edu/courses/cs143/

[40] "Software ImageJ" *Dipartimento di Informatica, Università degli studi di Bari*, 2008. http://www.di.uniba.it/~ig/GaborFilter/html/index.html

[41] **Loève, M.** "Probability theory", Vol. II, 4th ed.. *Graduate Texts in Mathematics. 46. Springer-Verlag*. ISBN 0-387-90262-7, 1978.

[42] **Harris, Zellig.** "Distributional Structure". *Word 10* (2/3): 146–62, 1954.

[43] **Li Fei-Fei, Rob Fergus, Antonio Torralba.** "Recognizing and Learning Object Categories", *Course on Recognizing and Learning Object Categories*. http://people.csail.mit.edu/torralba/shortCourseRLOC/

[44] **Wikipedia.** "Validación cruzada" http://es.wikipedia.org/wiki/Validación_cruzada

[45] **Lloyd, S. P.** "Least square quantization in PCM". Bell Telephone Laboratories Paper. Published in journal much later: Lloyd., S. P. (1982). "Least squares quantization in PCM". *IEEE Transactions on Information Theory 28* (2): 129–137, 1957.

[46] **Bosch, Anna; Zisserman, Andrew and Munoz, Xavier.** "Scene classification via pLSA". *Proc. 9th European Conference on Computer Vision (ECCV'06)* Springer Lecture Notes in Computer Science 3954: 517~530, 2006.

[47] **K. Grauman and T. Darrell.** "The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features". *Proc. of IEEE International Conference on Computer Vision*, 2005.

[48] **F. Perronnin, J. Sánchez and T. Mensink.** "Improving the Fisher Kernel for Large-Scale Image Classification". *ECCV 2010*.

[49] **A. Vedaldi and Zisserman.** "Efficient Additive Kernels via Explicit Feature Maps", in *Pattern Analysis and Machine Intelligence*, 34(3), 2012

[50] **Y-Lan Boureau, Francis Bach, Yann LeCun Jean Ponce.** "Learning Mid-Level Features for Recognition", *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'10)*, IEEE, 2010

[51] **Devijver, Pierre A. and Kittler, Josef.** "Pattern Recognition: A Statistical Approach", *Prentice-Hall, London*, GB, 1982.