UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

Detección automática de objetos en imágenes mediante el uso de puntos característicos

BADER AUDEH PIÑAR

SEPTIEMBRE 2012

Detección automática de objetos en imágenes mediante el uso de puntos característicos

AUTOR: Bader Audeh Piñar

TUTOR: Luis Fernando Lago Fernández

Grupo de Neurocomputación Biológica (GNB)

Dpto. de Ingeniería Informática

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Septiembre de 2012

Resumen

El principal objetivo de este proyecto de fin de carrera es la detección automática de objetos en imágenes mediante el uso de puntos característicos. Para ello se ha realizado un estudio del estado del arte en el que se han evaluado diferentes métodos para la clasificación de imágenes, utilizando diferentes bases de datos públicas. Uno de los métodos más extendidos en el estado del arte es el modelo '*Bag-of-words*', este método consiste en la generación de un diccionario visual o *'codebook'* con los "términos visuales" que aparecen en las imágenes y caracterizando así la imagen, creando un histograma que represente el número de ocurrencias de estos términos visuales. Finalmente, se clasifican las imágenes mediante el uso de un clasificador del tipo paramétrico o no paramétrico. En este proyecto se han estudiado tanto el método '*Bag-of-Words*' como otros modelos de clasificación con el fin de construir un sistema final utilizando el método más óptimo para la clasificación de imágenes. Pese a que nuestros resultados con el modelo BOW no son tan buenos como los vistos en la literatura, hemos diseñado un sistema final más sencillo que nos proporciona resultados muy satisfactorios dentro de un entorno controlado.

Palabras clave

Puntos característicos, SIFT, clasificación de imágenes, codebook, *Bag-of-Words*, k-means, Naïve-Bayes Nearest-Neighbor (NBNN), Support Vector Machine (SVM), Receiver Operating Characteritics (ROC), Precision/Recall (P/R).

Abstract

The main purpose of this final year project is the automatic object detection in images using keypoints. We have performed a study of the State of the Art evaluating different methods for image classification, using public databases. One of the most extended methods is the model called *Bag-of-Words*. This method is based on the generation of a codebook with the visual terms appearing in the images, and characterizing the images by a histogram that represents the number of occurrences of each of these visual terms. Finally, for the image classification we use parametric or non-parametric classifiers. In this project we have studied both the BOW model and other classifications models with the final purpose of building a final system using the best method for image classification. Although our results with BOW are not as good as in the literature, we have designed a final system that provides satisfactory results in a controlled environment.

Key words

Features points, SIFT, image classification, codebook, *Bag-of-Words*, k-means, Naïve-Bayes Nearest-Neighbor (NBNN), Support Vector Machine (SVM), Receiver Operating Characteritics (ROC), Precission/Recall (P/R).

Agradecimientos

En primer lugar quiero dar las gracias a mi tutor, Luis Fernando Lago, por su dedicación y apoyo durante la realización de este proyecto. Del mismo modo, también quiero agradecer a Richard Rojas por su ayuda a todas mis dudas y problemas que me han ido surgiendo a lo largo de este proyecto, y por todas esas horas de entretenimiento. Agradezco también a Ana Pérez Fernández y a Eduardo Cermeño Mediavilla por facilitar una de las bases de datos.

A todos los profesores que he tenido durante mi vida, tanto en el colegio como en la universidad, todos ellos me han ayudado mucho.

A Dani, por la gran amistad que hemos mantenido durante todos los años, desde la infancia hasta ahora y lo que nos queda y por todos esos grandes momentos que hemos pasado juntos.

A mi amigo Jesús Benítez, al que nunca olvidaré y al que siempre tendré en mi corazón, todos los días se te echa de menos!

A todos los amigos que he conocido en la universidad, Alberto, Alicia, Suse, Edu, Silvia, Sarita, María, Borja, Laura, por todos los grandes momentos que hemos vivido juntos y que viviremos, por todas esas fiestas y todos esos viajes que nos hemos pegado, espero que nunca se acaben.

También a todos ellos que he conocido fuera de la universidad y que ahora son una parte muy importante de mi vida, Sara, Lucas, Juan.

En especial, agradecer a Isa por todos estos años que hemos pasado juntos, tanto en la universidad como en el trabajo, por nuestras fiestas y nuestras vueltas a casa, ha sido y será siempre un placer tenerte como vecina.

Y por último, y el más importante durante todos estos años en la carrera, Álvaro, por tu humor y amistad, gracias por todo tu apoyo en todo este tiempo y gracias por estar cuando siempre te he necesitado, GRACIAS!

A Claudia, por ser como es, por sacarme una sonrisa siempre que te veo, y por hacerme ser feliz, gracias por aguantarme y por apoyarme siempre. ¡Eres lo más importante de mi vida! Gracias por todo este tiempo junto a ti y por todo lo que nos queda. Gracias también a la familia Molina por cuidarme siempre.

Gracias a toda mi familia. A Yahia por ser un buen hermano mayor que me ha cuidado siempre y que siempre ha confiado en mí. A Zoraida por ser la mejor persona que he conocido en mi vida, por aguantarme y por su cara siempre sonriente. A mi padre por enseñarme a ser la persona que soy y a no dejar nunca nada por imposible. Y en especial a mi madre, por su apoyo, por cuidarme siempre y por todo su cariño! GRACIAS!!!!

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1. Introducción

1.1 Motivación

La categorización automática de imágenes consiste en la asignación de una o varias etiquetas a una imagen en base a su contenido semántico. Es un campo cada vez más investigado con numerosas aplicaciones en áreas como la recuperación de imágenes basada en contenido [1], la seguridad y la video-vigilancia [2], el diagnóstico a partir de imágenes médicas [3], la visión y navegación de robots [4], etc.

El problema de la categorización automática de imágenes está actualmente abierto, aunque en los últimos años se ha avanzado mucho [5; 6; 7]. Hoy en día, y gracias también a que la tecnología ha avanzado en paralelo (cámaras fotos, cámaras de video-vigilancia, internet, etc.), podemos utilizarla para mejorar en áreas como buscadores de internet para encontrar objetos en fotografías cuya solución se basa en si la imagen contiene ciertos objetos o no en función de su contenido visual, y no en las anotaciones textuales colindantes [1]. También en áreas medicas como por ejemplo en la detección automática de células cancerígenas [8; 9]. O incluso en áreas más tecnológicas como es en la utilización de un robot para que este pueda localizarse a sí mismo y construir un mapa del medio que le rodea [10].

El planteamiento más popular, está basado en el modelo *Bag-of-Words* (*BOW*) usado en la categorización de textos, y se conoce como *bag-of-visual-words*. En una representación del modelo *BOW*, un documento es descrito como un histograma que representa el número de ocurrencias de cada palabra [11]. De una forma parecida, una imagen se puede caracterizar mediante un histograma de ocurrencias de las palabras visuales que aparecen en ella. Las palabras visuales se refieren a características básicas como esquinas, bordes, etc.

El objetivo principal de este proyecto es el estudio de algoritmos para la clasificación automática de imágenes y su aplicación a un problema real. Para ello, abordaremos el problema utilizando bases de datos públicas utilizadas en la literatura [12; 5], y tras haber evaluado los algoritmos, aplicaremos los mismos métodos en un problema real propuesto por una empresa dedicada a la video-vigilancia. La Ilustración 1 nos muestra un ejemplo de imagen que puede encontrarse en cualquiera de las bases de datos, donde residen varios objetos, aunque nuestro trabajo consistirá únicamente en detectar uno de los objetos.

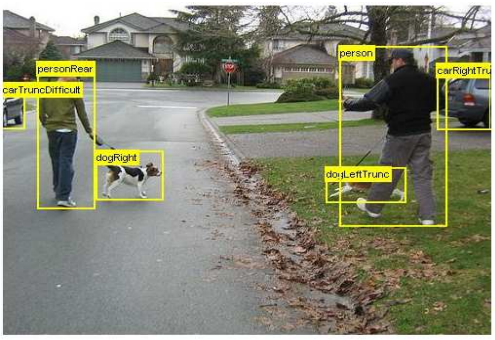


Ilustración 1: Ejemplo de imagen extraída de la base de datos de Pascal 2007, en la que se muestra mediante una serie de recuadros los objetos de interés para categorizar la imagen.

1.2 Objetivos

En este proyecto de fin de carrera se plantean dos objetivos principales:

- El estudio y la evaluación de algoritmos para la clasificación automática de imágenes en base a la presencia o no en ellas de determinados objetos, utilizando características de bajo nivel que han sido extraídas de las imágenes. Para ello, usaremos dos bases de datos públicas extraídas de la literatura [12; 5].
- Diseñar un sistema basado en los algoritmos estudiados para resolver un problema específico planteado por una empresa dedicada a la video-vigilancia. Este problema consiste en clasificar como coches o autobuses los objetos que se extraen de las imágenes proporcionadas por uno de sus sistemas de vigilancia.

Para alcanzar los objetivos que se plantean en este proyecto, se llevarán a cabo las siguientes tareas:

- **Estudio del estado del arte:** En primer lugar, se hará un estudio de los diferentes algoritmos utilizados para la detección de objetos mediante la caracterización de las imágenes, así como, un estudio de los diferentes métodos de extracción de puntos de interés y de los algoritmos de clasificación.
- **Implementación de métodos de clasificación:** Se implementarán los métodos de clasificación *Bag-of-Words + Support Vector Machine* (BOW [13]+SVM [14]) y *Naive-Bayes Nearest* Neighbor (NBNN [6]). En ambos métodos se utilizarán puntos de interés extraídos mediante el algoritmo SIFT [15].
- **Evaluación:** Se evaluarán los algoritmos implementados utilizando dos bases de datos obtenidas de la literatura, tanto Pascal 2007 [5] como la base de datos extraída de [12]. Los resultados se representarán por medio de curvas de *Precisión/Recall* [16], y curvas ROC, (*Receiver Operating Characteristic)* [17] para poder comparar los resultados con los reportados en la literatura.
- **Diseño del sistema final:** Se creará un sistema para la detección de objetos en un entorno más específico. Para ello, se utilizará una base de datos privada proporcionada por una empresa de seguridad. El sistema consistirá en la caracterización de las imágenes y posteriormente en la aplicación del método de clasificación de imágenes que mejores resultados nos haya proporcionado anteriormente.
- **Análisis de resultados y conclusiones:** Se analizaran los resultados obtenidos con la nueva base de datos, utilizando las mismas curvas y se extraerán las conclusiones finales.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1:** En primer lugar, se hará una breve introducción a los sistemas de clasificación. Se describirá la motivación de este proyecto y finalmente comentaremos los objetivos principales de este trabajo.

- **Capítulo 2:** Estudio del estado del arte de los algoritmos que se utilizan en la literatura para la detección de imágenes mediante la extracción de puntos de interés y los métodos de clasificación más utilizados.

- **Capítulo 3:** Diseño e implementación de los algoritmos utilizados para alcanzar tasas favorables en cuestión a la detección de objetos en las imágenes**.**

- **Capítulo 4:** Pruebas, resultados experimentales y análisis comparativo entre los diferentes métodos y los resultados reportados en VOC PASCAL 2007 [5], bases de datos privada y bases públicas que se encuentren en la literatura.

- **Capítulo 5:** Finalmente, se discutirán los resultados obtenidos y se comentarán las conclusiones de los resultados. Por último, se propondrá una serie de trabajos futuros.

2. Estado del arte

2.1 Introducción

La categorización de imágenes se puede plantear como un problema de clasificación de patrones que consiste en asignar una o varias etiquetas a una imagen en base a su contenido semántico. Uno de los planteamientos más frecuentes consiste en modelar la distribución de características de bajo nivel contenidas en las imágenes, sin tener en cuenta las posiciones absolutas o relativas de dichas características. La recuperación de imágenes basada en su contenido (no semántico) mejoraría significativamente la calidad de las búsquedas. Para ello, es necesario disponer de modelos que se enfrenten a la clasificación de una imagen a partir de las características extraídas de las imágenes.

Debido a que la categorización de imágenes es un problema bastante complicado de resolver existen múltiples algoritmos y grupos de investigación dedicados a estudiar este problema y a dar posibles soluciones.

Estos grupos de investigación siguen un procedimiento general para intentar dar con una solución óptima para este tipo de problema que se puede dividir en una serie de pasos:

- Descripción de las imágenes en base a su contenido. Esta descripción se realiza normalmente mediante características de bajo nivel como color, textura, bordes, etc.
- Debido a la alta dimensionalidad de estas características, el siguiente paso es la reducción de la dimensión de las características extraídas.
- Clasificación de las imágenes mediante los algoritmos adecuados.

Uno de los modelos más utilizados para la categorización visual genérica que sigue el esquema descrito anteriormente, es el modelo llamado '*Bag-of-Words'*. Este modelo consiste en la generación de un diccionario visual o *'codebook'* con los "términos visuales" que aparecen en las imágenes y caracterizando así la imagen mediante un histograma en el que se representen el número de ocurrencias de los términos visuales iguales dentro de la imagen.

La caracterización de la imagen se puede realizar de diferentes maneras. El extractor más utilizado es el descriptor SIFT [18], aunque también se utilizan detectores de bordes y esquinas [19], el detector de SUSAN [20], Geometric-Blur (GB) [21], etc.

Debido a que el número de características diferentes extraídas con estos métodos es grande, se suelen cuantizar para poder así generar '*codebooks'* más pequeños con los que poder obtener representaciones de las imágenes más compactas. El problema de la cuantización es la pérdida de información que puede ser relevante para la solución. Existen métodos de aprendizaje que intentan compensar este problema [6]. El método más utilizado es el algoritmo k-means [22] aunque también se utilizan otros algoritmos como el Modelo de Mezcla de Gaussianas (GMM) [23], *mean-shift* [24], o árboles de decisión [25] .

Una vez generados estos '*codebooks'*, se procede a la clasificación de las imágenes con métodos propios del aprendizaje automático como SVM [14], K-Nearest Neighbor (K-NN) [26] o el clasificador pLSA (*probabilistic Latent Semantic Analysis*) [26; 27]. Otro método para la clasificación de imágenes es el llamado NBNN [6], que a diferencia del modelo BOW, no requiere de una fase previa de entrenamiento/aprendizaje mediante algoritmos como k-means o GMM. Este método de clasificación detecta el objeto mediante un cálculo de distancias entre el objeto la clase más cercana (la más similar) en la base de datos. Otro clasificador que proporciona buenos resultados en la detección de objetos es el llamado *Stacked R. Forest* [28].

Una vez clasificadas las imágenes, se procede finalmente a la evaluación del clasificador, normalmente mediante el uso de gráficas de Precision/Recall o curvas ROC, calculando la Precisión Media del clasificador y el Área bajo la Curva, en inglés '*Area Under Curve*' (AUC) de la curva ROC.

En los siguientes apartados describiremos el método BOW de clasificación de imágenes y las técnicas más utilizadas en la literatura para resolver el problema de clasificación de las imágenes.

2.2 Bag-Of-Words (BOW)

El modelo *bag-of-words* es un método muy utilizado en la categorización de texto. En esta categorización, el documento es tratado como un histograma en el que se cuenta el número de ocurrencias de cada palabra. Siguiendo el mismo procedimiento que en la categorización de textos, se tratan las imágenes como documentos y se crea un histograma que cuente el número de ocurrencias de ciertas características (puntos SIFT, detectores de borde y esquinas, colores, etc.) de la imagen.

La implantación del modelo BOW en la visión artificial requiere los siguientes pasos: detección de características, descripción de características y generación de *codebooks*.

2.2.1 Extracción de puntos de interés

Como ya hemos comentado en la introducción, antes de generar los '*codebooks*' se obtienen una serie de características de bajo nivel de todas las imágenes. Estas características también llamadas puntos característicos o puntos de interés caracterizarán las imágenes.

Los detectores de puntos de interés tratan de encontrar características como bordes, esquinas, color, etc. Algunos de estos detectores famosos son el detector de esquinas de Harris [19], el detector SUSAN (Smallest Univalue Segment Assimilating Nucleus) [20], el detector SURF (Speeded-Up Robust Features) [29], el detector de puntos característicos SIFT de Lowe [18]. Para la extracción de puntos de interés se ha escogido el detector SIFT debido a que es un descriptor invariante a posibles cambios de escala, traslación, intensidad y orientación. .

Después de la detección de características, cada imagen es representada a través de sus características locales. Los métodos de representación tratan de describir estas características como vectores numéricos, llamados descriptores de características. El descriptor debe tener la habilidad de manejar la intensidad, rotación, escala y variaciones afines de la misma dimensión.

En la Ilustración 2 podemos observar como diferentes detectores de puntos de interés, caracterizan una imagen de diferente manera.

A continuación describiremos algunos detectores de puntos de interés, principalmente el detector SIFT ya que es el más utilizado y el que usaremos en este proyecto.

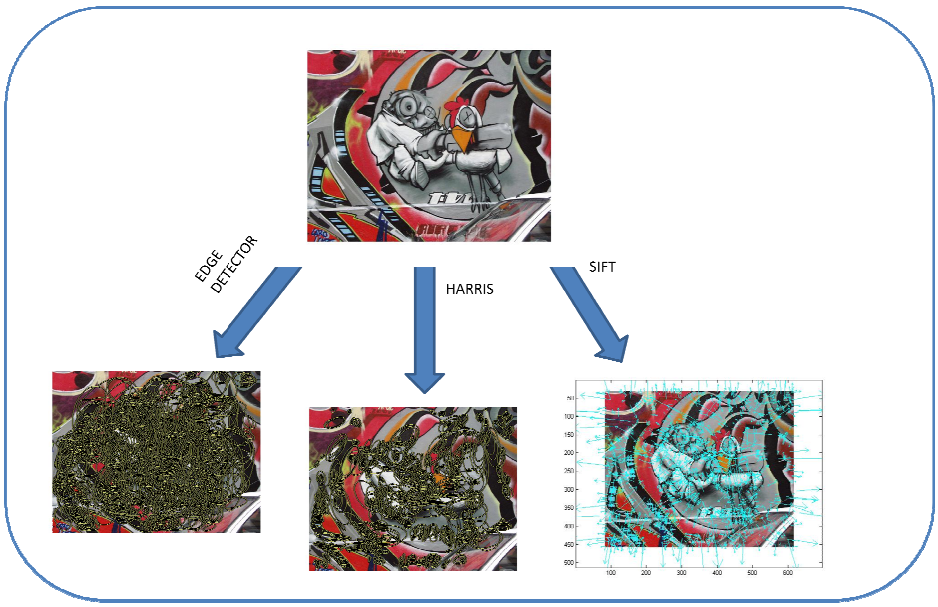


Ilustración 2: Aplicación de puntos de interés en la imagen, la imagen superior es la original, la imagen de la izquierda tras utilizar el detector de bordes descrito en el artículo [19], la imagen central es el resultado tras haberle aplicado el detector de esquinas de Harris [30] y la imagen de la derecha es el resultado tras haberle aplicado el detector SIFT [31]

- Scale Invariant Feature Transform (SIFT)

SIFT (Scale Invariant Feature Transform) [18] es un algoritmo de visión artificial que permite detectar y, posteriormente describir características en regiones locales de una imagen de una forma invariante a escala. Para generar un conjunto de características sobre una imagen el algoritmo utiliza:

- **Detección de extremos en el espacio-escala**

  El primer paso para detectar puntos de interés o puntos clave de una imagen es identificar localizaciones y escalas que se repiten continuamente usando diferentes vistas del mismo objeto. Para detectar estas localizaciones, que son invariantes a diferentes escalas de la imagen, se buscan características estables a través de todas las escalas. Para ello se utiliza una función continua de la escala conocida como espacio de escala. Al espacio de escala lo vamos a definir como una función de espacio (ejes de coordenadas) al que se le añade una nueva coordenada que va a ser la escala.

  El espacio de la escala se define como una función, $L(x, y, \sigma)$, que se obtiene tras convolucionar la función Gaussiana $G(x, y, \sigma)$ con la imagen $I(x, y)$.

  Donde $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$.

  Para que la detección de los puntos clave sea eficaz, el algoritmo utiliza la diferencia de Gaussianas (DoG) convolucionada con la imagen. Esta diferencia puede ser obtenida a partir de las diferencias entre dos escalas vecinas separadas por un factor k que es constante. La función de detección queda de la siguiente manera:

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma).$$

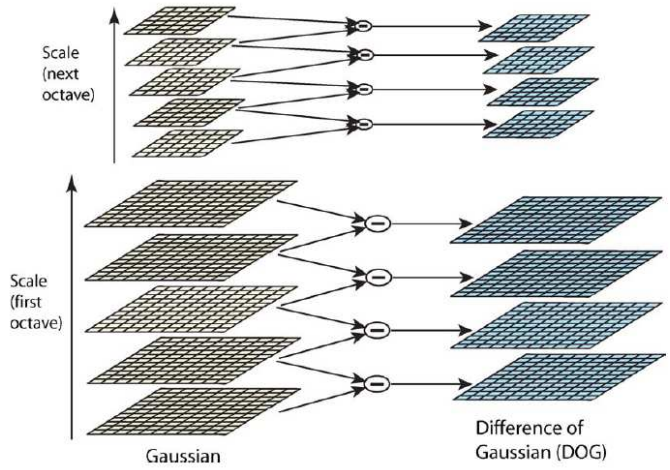


Ilustración 3: Aproximación eficiente para la construcción de la función de detección. La imagen inicial es convolucionada continuamente con Gaussianas (variando k) para crear el grupo de imágenes mostradas a la izquierda. Las imágenes de las Gaussianas vecinas se restan para producir las imágenes de Diferencia de Gaussianas que hay a la derecha. A continuación se repite el mismo proceso pero variando la escala (σ).

- **Localización de los puntos de interés**
  De todos los puntos hallados anteriormente se tendrán que eliminar aquellos que no sean relevantes, por lo que a cada punto se le aplica un modelo para saber su localización y escala. Para averiguar los máximos y mínimos locales de *D(x, y,σ)*, cada punto se compara con sus ocho píxeles vecinos en la imagen actual y con los nueve vecinos de las imágenes superior e inferior, en total con los 26 puntos vecinos, como observamos en la Ilustración 4. Así seleccionamos únicamente aquellos puntos que sean, o más grandes que todos ellos o más pequeños.

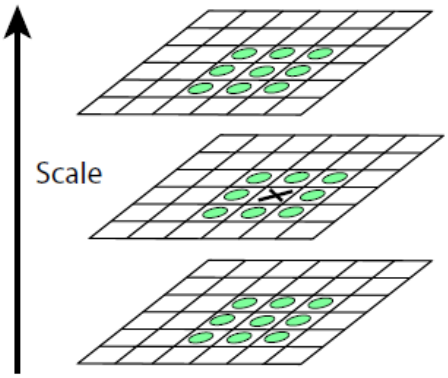


Ilustración 4: Comparación de pixel con sus ocho vecinos y los nueve vecinos de las escalas superior e inferior

Estos puntos clave (máximos y mínimos locales descritos en el punto anterior) serán candidatos a puntos característicos. El siguiente paso es eliminar aquellos puntos que tengan un menor contraste, es decir que son sensibles al ruido y por lo tanto son inestables en imágenes con baja resolución. Para ello se descartan los puntos extremos mediante una umbralización.

Dado que la función Diferencia de Gaussianas devuelve muchos puntos clave a lo largo de bordes y esquinas de los objetos, habrá que eliminarlos para mantener la estabilidad de los puntos (extremos). Para ello se calcularan las principales curvas a través de una matriz Hessiana, H, calculada en la localización y escala de los puntos clave (máximos y mínimos).

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Los autovalores de H son proporcionales a estas principales curvas de D. Si utilizamos la aproximación de [19], podemos evitar el cálculo de los autovalores ya que lo único que nos concierne es el ratio de estas curvas. Si definimos α como el autovalor con la magnitud más grande y β como el autovalor con la magnitud más pequeña, se podrá calcular la traza como la suma de los autovalores de la diagonal principal de *H* y su producto como el determinante de *H*.

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$Det(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta$$

Si el determinante es negativo, las curvas son de diferente signo por lo que el punto es descartado por no ser un extremo. Si *r* es el ratio entre el autovalor más grande y el más pequeño, por lo que α=rβ. Entonces,

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r+1)^2}{r}$$

Por lo que, para comprobar si el ratio de las principales curvas es menor que un umbral especifico, r, solo es necesario comprobar si,

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

El valor de $\frac{(r+1)^2}{r}$ es mínimo cuando los autovalores son iguales y va creciendo según va aumentado el valor del ratio *r.* Si el ratio es mínimo, se considerará que el punto pertenece a un borde.

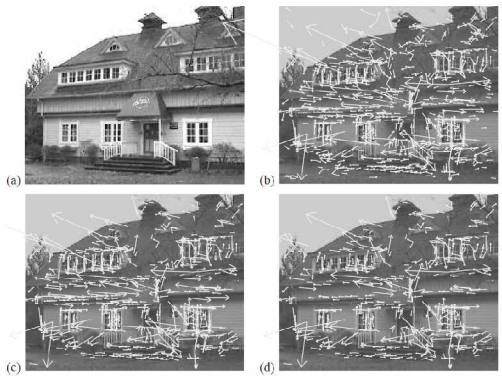


Ilustración 5: Fases de selección de puntos clave a) la imagen original con una baja resolución (233x189) píxeles, b) los 832 puntos clave originales, máximos y mínimos de la función Diferencia de Gaussianas, c) 729 puntos clave restantes tras aplicar un umbral con un mínimo

contraste, y d) los 536 puntos clave finales que quedan siguiendo un umbral adicional en proporción a las principales curvaturas.

- **Asignación de la orientación**
  Una vez excluidos aquellos puntos que son sensibles al ruido o se encuentran en bordes o esquinas de los objetos, se pasa a asignar una orientación a cada punto característico.
  Para cada imagen *L(x, y)* con una determinada escala, se calcula la magnitud del gradiente *m(x, y)* y la orientación *θ(x, y)* mediante diferencias entre pixeles:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}$$

- **Descriptor del punto de interés**
  Las anteriores operaciones nos han proporcionado a cada punto una localización, una orientación y una escala, por lo que los puntos característicos son invariantes a estos parámetros. El siguiente paso tratara de calcular un descriptor para las regiones locales de la imagen y conseguir así una mayor invariancia a otros parámetros, como puede ser la iluminación. Para conseguir el descriptor de cada punto, es necesario obtener las magnitudes y orientaciones de los vecinos del punto de interés, utilizando la imagen suavizada más cercana a la escala calculada en el segundo paso. Una vez obtenidos, para asegurar la invariancia a la orientación, las coordenadas del descriptor y las orientaciones del gradiente se rotan de forma relativa a la orientación extraída antes.

Todo este proceso nos da como resultados una serie de puntos característicos, y cada uno de ellos con un descriptor final de 128 componentes, que son invariantes a traslación, escala, orientación y parcialmente a cambios de intensidad.

Un aspecto importante de esta aproximación es que genera un gran número de características que cubre densamente la imagen sobre el rango completo de escalas y localizaciones. Con una imagen típica de 500x500 píxeles se pueden producir unas 2000 características estables.

Existen múltiples detectores además de SIFT para la extracción de características de una imagen. Los más conocidos son SURF (variación del SIFT que acelera los cálculos de los puntos característicos), el detector de esquinas y bordes SUSAN o el detector Harris. Debido a que los puntos SIFT, son puntos característicos invariantes a los parámetros vistos, (localización, escala, orientación y a iluminación) y a que presentan un coste computacional relativamente pequeño. Además de utilizarse en detección de objetos, es muy utilizado en otras aplicaciones como localizador para un robot con imágenes captadas desde su cámara, para 'mapear' [10], reconocimiento de imágenes [4], reconstrucción de imágenes en 3D [27], *tracking*, etc., y otras aplicaciones cuya finalidad sea la identificación de imágenes entre un conjunto.

- Speeded Up Robust Features (SURF)

SURF (Speeded Up Robust Features) [29] es otro de los algoritmos junto a SIFT mas utilizados en la literatura para la extracción de puntos de interés invariantes. Este algoritmo es relativamente parecido al propuesto en SIFT. Es un algoritmo con una gran robustez ante transformaciones de la imagen. Mejora la velocidad de procesado del detector SIFT reduciendo la dimensión y la complejidad del descriptor, de modo que los puntos continúen siendo suficientemente característicos e igualmente repetitivos.

Antes de hallar la orientación, es necesario el uso de un detector para la extracción de puntos de interés, en este caso se utiliza el detector Fast Hessian.

El primer paso consiste en hallar la orientación del punto de interés partiendo de una región circular alrededor del punto. Después se construye una región cuadrada alineada con la orientación y se extrae el descriptor de ahí.

El algoritmo SURF utiliza una aproximación básica de la matriz Hessiana para reducir el tiempo de computación [29]. La matriz Hessiana se utiliza debido a su buena relación entre la precisión y el coste temporal. Dado un punto X=(x, y) y una imagen I, la matriz Hessiana $\mathrm{H}(\mathrm{X}, \sigma)$ en X con una escala σ se define de la siguiente forma:

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

Para localizar estos puntos, el máximo del determinante de la matriz H es interpolado en el espacio-escala y en el espacio de la imagen con el método propuesto en [32]. Una vez obtenidos los puntos de interés, se calcula su orientación, mediante el cálculo de la respuesta *Haar* en la dirección *x* e *y* en un área circular alrededor del punto.

Una vez obtenida la orientación se extrae el descriptor, para ello se construye una región cuadrada alrededor del punto y orientada en relación con la orientación calculada en el paso anterior. La región cuadrada será reducida progresivamente en regiones más pequeñas 4x4 como vemos en la Ilustración 6.

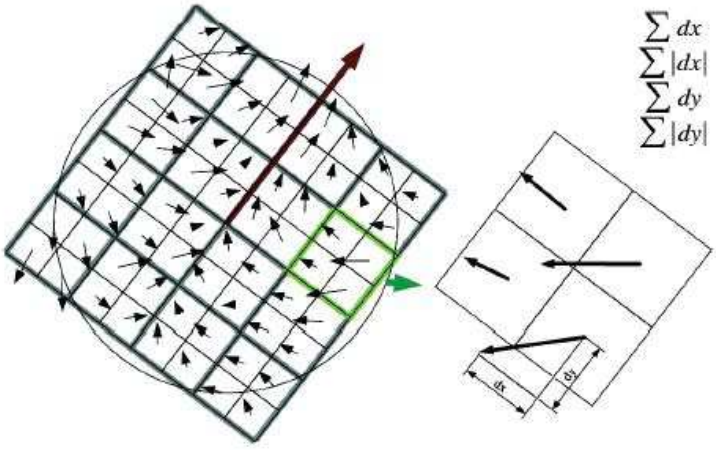


Ilustración 6: Para construir el descriptor, tal y como se muestra en la imagen de la izquierda se toma una cuadricula orientada de 4x4 sub-regiones cuadradas sobre el punto de interés.

En SIFT [18] a diferencia de SURF, los datos extraídos contienen la localización, la escala y la orientación, puesto que es posible que en una misma posición *(x, y*) encontremos varios puntos de interés con distinta escala y/u orientación. En cambio, en SURF, en una misma posición

aparece un único punto de interés, por lo que no guarda la escala y la orientación, obteniendo al final un total de 64 componentes.

- Smallest Univalue Segment Assimilating Nucleus (SUSAN)

SUSAN (Smallest Univalue Segment Assimilating Nucleus) [20] es un algoritmo que permite procesar imágenes de bajo nivel, principalmente detectando esquinas y bordes. Para la detección de características, SUSAN sitúa una máscara circular sobre el pixel que se va a analizar (el núcleo). Cada pixel es comparado con el núcleo usando la función de comparación:

$$c(\vec{m}) = e^{-\left(\frac{(I(\vec{m})-I(\vec{m}_0))}{t}\right)^6},$$

Donde I es la intensidad, t determina el radio, $\vec{m}$ es un pixel perteneciente a la máscara circular M, $\vec{m}_0$ es el núcleo y el valor del exponente ha sido calculado empíricamente.

El área de SUSAN es dada por:

$$n(M) = \sum_{\vec{m}\in M} c(\vec{m}),$$

Y la respuesta del operador SUSAN es obtenida mediante la fórmula:

$$R(M) = \begin{cases} g - n(M) & if\ n(M) < g \\ 0 & otherwise, \end{cases},$$

Donde el valor de *g* determina el tamaño mínimo del segmento. Si *g* es lo suficientemente grande, se convierte en un borde.

- Detector Harris

El detector de Harris [33], es un detector de esquinas. Está basado en la denominada matriz de auto-correlación, muy utilizada en la detección de características o de estructuras locales en imágenes. Esta matriz debe ser adaptada a cambios de escala para hacerla independiente de la resolución de la imagen. Se define como:

$$\mu(x,\sigma_1,\sigma_D) = \begin{bmatrix}\mu_{11} & \mu_{12}\\ \mu_{21} & \mu_{22.}\end{bmatrix} = \sigma_D^2 g(\sigma_1) * \begin{bmatrix} L_x^2(x,\sigma_D) & L_xL_y(x,\sigma_D)\\ L_xL_y(x,\sigma_D) & L_y^2(x,\sigma_D)\end{bmatrix}$$

Donde σ_1 es la escala, σ_D es la diferenciación de la escala y L es la derivada calculada en la dirección del punto (x, y). Para la detección de esquinas combina la traza y el determinante de esta matriz, siendo esquinas los máximos locales de la función:

$$esquina = \det\big(\mu(x,\sigma_1,\sigma_D)\big) - k * trace^2(\mu(x,\sigma_1,\sigma_D))$$

Los últimos dos detectores descritos (SUSAN y Harris) se tratan de algoritmos que sólo detectan puntos característicos, es decir, nos dan la localización del punto característico en sí, no un descriptor del punto como se obtienen con SIFT o SURF. Aunque también se puede utilizar SIFT como descriptor únicamente, habiendo localizado el punto anteriormente mediante otro algoritmo, a pesar de que para ello se pierda la invarianza frente a la traslación y el cambio de escala del punto, esta forma de utilizar SIFT es ampliamente utilizado [7].

2.2.2 **Reducción de dimensionalidad**

Los descriptores de los puntos característicos que hemos visto anteriormente presentan una alta dimensionalidad (128 componentes en el caso de SIFT y 64 componentes en el caso de SURF). Por ello, en la mayoría de ocasiones, es necesario aplicar un método de reducción de dimensionalidad

para así eliminar información poco relevante de los vectores de características que puede llegar a actuar como distractor [5; 7].

En la mayoría de los sistemas estudiados, la reducción de dimensionalidad se realiza mediante análisis de componentes principales (Principal Component Analysis, PCA) [34]. El análisis de componentes principales es una técnica estándar para reducir las dimensiones de un conjunto de datos y ordenarlas por importancia. El funcionamiento técnico de PCA se basa en buscar la proyección según la cual los datos queden mejor representados en términos de mínimos cuadrados. Los nuevos componentes o factores principales serán una combinación lineal de las variables originales y además serán linealmente independientes entre sí.

Este método construye una transformación lineal que elige nuevos ejes de coordenadas, donde la dirección de mayor varianza del conjunto de datos coincide con el primer eje de coordenadas, el segundo, con la de mayor varianza y así sucesivamente. Un método estándar para desechar los ejes con menor varianza es la de ir seleccionando los ejes de más a menos varianza hasta que la suma de los que hemos seleccionado llegue a la varianza total que se quiere capturar.

Sean *X* vectores de *n* dimensiones, PCA pretende encontrar un número de factores *p<n* que expliquen aproximadamente el valor de las *n* variables de cada sujeto. El hecho de que existan estos p factores puede interpretarse como una reducción de la dimensión en la que antes se necesitaban n dimensiones para definir un vector y ahora solo basten *p* valores para definirlo.

Se define F_j como el conjunto de variables aleatorias. A partir de estos datos se puede construir la matriz de correlación muestral:

$$\mathcal{R} = [r_{ij}] \in N_{nxn}, \text{ donde } r_{ij} = \frac{cov(F_i, F_j)}{\sqrt{var(F_i)}\sqrt{var(F_j)}}.$$

Y puesto que la matriz de correlación es simétrica, es diagonalizable y se verifica que $\sum_{i=1}^{n} \lambda_i = 1$, siendo λ_i los valores de la matriz y los pesos de cada uno de los *n* componentes principales. Los factores principales identificados matemáticamente se representan por la base de autovectores de la matriz $\mathcal{R}$. Cada una de las variables puede ser expresada como la combinación lineal de sus componentes principales. Un ejemplo de cómo funciona PCA se puede observar en la Ilustración 7.

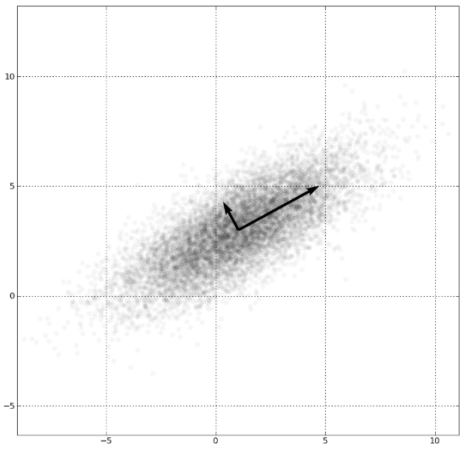


Ilustración 7: Ejemplo de PCA de una distribución normal multivariante. Los vectores muestran los autovectores de la matriz de correlación.

2.2.3 **Generación de codebooks**

Una vez caracterizadas todas las imágenes, mediante la extracción de puntos característicos, y observando que no todas las imágenes tienen el mismo número de puntos de interés se suele realizar el denominado diccionario visual, o vocabulario representativo de las características que aparecen. A este vocabulario se le denomina '*codebook*' y sirve para describir las imágenes usando un mismo número de características, agrupando estas características en un mismo grupo denominado '*cluster*'. La construcción de un *codebook* se puede realizar de diversas formas. Típicamente, los métodos más utilizados para generar los *clusters* son algoritmos de agrupamientos como *k-means* [35], en el que la asignación se realiza mediante la cercanía del punto característico al *cluster*, o el algoritmo GMM (*Gaussian Mixture Model*) [23], en el que la asignación se realiza mediante la probabilidad de que un punto pertenezca a ese *cluster*. Una vez elaborado el *codebook* con k características, cada imagen se caracteriza como un vector de k componentes, donde la componente k es el número de apariciones de la característica k en esa imagen.

- Algoritmo K-means

El algoritmo de *clustering k-means* [35] está basado en un criterio de agrupamiento cuya idea principal es definir un número de centroides k, el cual será elegido empíricamente, uno por cada cluster deseado. Estos centroides deben ser situados inicialmente del mejor modo posible, ya que la situación inicial de los centroides condicionará el resultado. El siguiente paso es tomar cada punto perteneciente a un conjunto de datos y asociarlo al centroide más cercano. Cuando no queda ningún punto por asociar se habrá completado la primera iteración del algoritmo, y se habrá realizado un agrupamiento temprano o primer agrupamiento. En este punto, se necesita volver a calcular *k* nuevos centroides a partir de los clusters resultantes del paso previo. Este proceso se repite de forma iterativa hasta que converge y no se puedan realizar más cambios, es decir no mejoran las situaciones de los cluster, o el bucle implementado se termina por alguna restricción puesta por el usuario.

En otras palabras, el algoritmo divide los datos **x** incluidos en un conjunto **X** en **k** conjuntos de *clusters* $\mathbf{S_1, S_2,\ldots S_k,}$ minimizando la distancia, normalmente la distancia Euclidea (aunque existen otras distancias como el arco coseno), entre **x** y la media de los datos pertenecientes a $\mathbf{S_1}$. En este método los datos que están cerca de la media son agrupados al cluster **i**.

Resumiendo, para la elección de los clusters hay que seguir estos cuatro pasos:

1.-Eleccion de los k centros o centroides de los clusters iníciales.

2.- Asignación de los datos a cada cluster.

3.-Elección de nuevos centroides.

4.-Repetimos el proceso a partir del paso 2, hasta que no varíen los centroides.

- Modelo de mezcla de Gaussianas

Un modelo de mezcla de Gaussianas (GMM) [23] es una función de densidad de probabilidad paramétrica que se representa como la suma de los pesos de las componentes de densidades Gaussianas. GMM se usa normalmente como un modelo paramétrico de las distribuciones de probabilidad en medidas continuas. Para el cálculo de GMM hay que ajustar una serie de parámetros como la matriz de covarianza, la media y el peso de cada componente de la mezcla de las Gaussianas. Para ello, dados una serie de puntos, habrá que estimar estos parámetros a través del método de máxima verosimilitud o en inglés, '*maximum likelihood*' (ML). Para conseguir la

estimación de los parámetros mediante ML se utiliza el algoritmo llamado *'expectation-maximization'* (EM) [36].

La idea principal del algoritmo EM es la siguiente: dado un modelo inicial con un conjunto de parámetros λ, estimar otro modelo (con parámetros $\bar{\lambda}$) en el que $p(X|\bar{\lambda}) \geq p(X|\lambda)$, donde *p* es la probabilidad de ocurrencia de los datos observados. Esto es lo que se denomina verosimilitud '*likelihood'* del modelo. El nuevo conjunto de parámetros se convierte en el inicial y se repite el proceso hasta que la solución converge.

Una vez que el algoritmo converge cada componente de la mezcla se considera como un *cluster* y se van asignando los puntos a cada *cluster* según la probabilidad a la que pertenezcan a cada *cluster*.

2.2.4 Aprendizaje automático para clasificación

Una vez descritas todas las imágenes con el mismo número de características (mediante los *codebook*), se llega al último paso, la clasificación. Para ello se pueden utilizar diferentes métodos de clasificación automática.

El aprendizaje automático trata el diseño y desarrollo de algoritmos que permiten a los ordenadores mejorar su rendimiento a la hora de analizar datos. Un mayor enfoque en la investigación del aprendizaje automático produce modelos, reglas y patrones de los datos. Como el conjunto de datos con el que se construye el modelo para clasificar (entrenamiento) es finito, la teoría de aprendizaje normalmente no da garantías absolutas sobre el rendimiento de los algoritmos. Existen diversos algoritmos de aprendizaje automático como las Máquinas de Vectores de Soporte (SVM) [37], redes neuronales artificiales (ANN), los arboles de decisión, etc. Estos clasificadores también son conocidos como paramétricos ya que necesitan de un entrenamiento previo para ajustar los parámetros de aprendizaje al clasificador.

Existen otro tipo de clasificadores denominados no paramétricos, que no necesitan de aprendizaje ó entrenamiento previo de parámetros como NBNN [6] ó VNN [38; 6]. En principio se puede utilizar cualquier clasificador para resolver el problema de clasificar imágenes, aunque el más utilizado es el SVM que son los que se van a describir a continuación.

- Las máquinas de vectores de soporte

La máquina de vectores de soporte o SVM (*Support Vector Machine*), [39; 40; 41] es un tipo de clasificador que utiliza hiperplanos para la clasificación de los datos. Para poder resolver un problema que no es linealmente separable se tendrán que proyectar los datos a un espacio con más dimensiones previamente. Usa un método óptimo para resolver problemas de aprendizaje automático y puede ser utilizado para clasificación de patrones y para regresión.

La idea principal es construir un hiperplano como una superficie de decisión en un espacio de características de alta dimensión en el cual se maximice el margen de separación entre las diferentes clases, ver Ilustración 8.

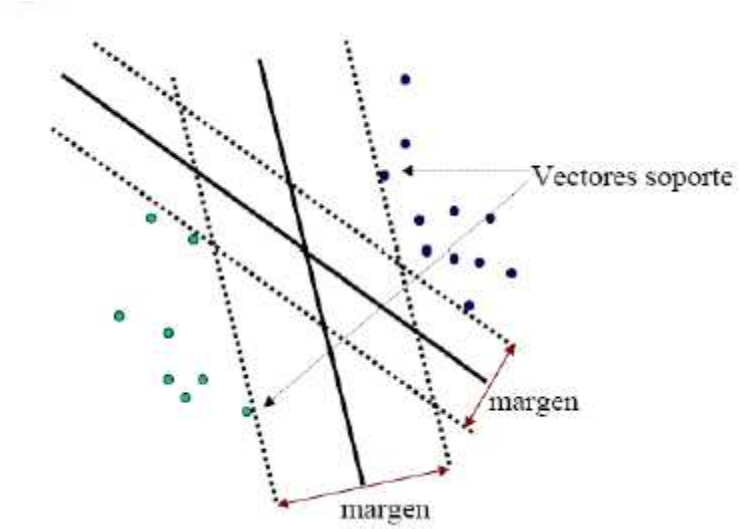


Ilustración 8: Idea del hiperplano óptimo para patrones linealmente separables

Para un vector de pesos *w* y un umbral *b* dados, la ecuación del hiperplano se definiría como $w^T\, x + b = 0$. La separación entre el hiperplano y los datos más cercanos se denomina margen de separación.

El objetivo de la máquina de vector de soporte es encontrar un hiperplano en particular tal que se maximice el margen al tiempo que se minimiza el error de clasificación sobre el conjunto de datos de entrenamiento. Si esto se cumple, la superficie de separación se denomina hiperplano óptimo.

Si consideramos el caso de patrones no separables, no va a ser posible construir un hiperplano separador sin encontrar errores de clasificación. De cualquier modo, se desea encontrar un hiperplano óptimo que minimice la probabilidad de errores de clasificación, promediado sobre el conjunto de entrenamiento. Esta idea se representa en la Ilustración 8. Se puede decir que el problema queda planteado de la siguiente manera: dado un conjunto de entrenamiento $\{(x_i, d_i)\}_{i=1}^{N}$ de *N* datos, hay que hallar los valores óptimos del vector de pesos *w* y del umbral *b* tal que satisfaga las restricciones:

$$d_i(w^T x_i + b) \geq 1 - \xi, \text{ para } i = 1,2,\dots,N, \xi_i \geq 0 \text{ para todo i}$$

Y tal que el vector de pesos *w* y las variables lentas o de holgura ξ_i minimicen la función de coste:

$$J(w,\xi) = C\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|w\|^2$$

donde el primer término de la ecuación tiene en cuenta el error y el segundo término el margen de separación (complejidad). La constante C es un parámetro positivo especificado por el usuario. El parámetro C controla el compromiso entre la complejidad de la máquina y el número de puntos no separables, puede ser visto como un parámetro de regularización frente a los errores de clasificación:

- C grande favorece modelos con menos error
- C pequeño favorece modelos más simples

Para problemas no lineales se realiza, antes de la construcción del hiperplano separador de máximo margen, una proyección no lineal de los datos sobre un espacio de muchas dimensiones. Así la probabilidad de encontrar una solución lineal es mayor (Teorema de Cover). Para construir diferentes proyecciones se utilizan diferentes tipos de *kernel*, de tipo polinomial, radial, etc.

El problema es que a mayor número de dimensiones más fácil será separar estos puntos pero el coste será más grande, por lo que habrá que equilibrar el número de dimensiones con el coste computacional del problema.

- Clasificación no paramétrica

Los clasificadores no paramétricos son aquellos que no necesitan de un proceso previo de aprendizaje/entrenamiento de parámetros. Uno de los más conocidos es el llamado *Nearest-Neighbor* (NN) [42]. Este método utiliza un algoritmo muy sencillo en el que clasifica un patrón mediante un cálculo de distancia, en el cual el patrón pertenece a la clase del patrón más cercano. En algunas ocasiones se utiliza K-NN, una variante de NN pero en vez de utilizar el patrón más cercano utiliza k patrones cercanos. Otro método utilizado es el método llamado *Parzen-Window* [43] que aproxima un conjunto de datos de entrenamiento mediante una combinación lineal de núcleos centrados en los puntos observados.

2.3 Clasificador Naive-Bayes Nearest Neighbor (NBNN)

El clasificador NBNN [6] se basa en un algoritmo que utiliza puntos característicos al igual que el modelo BOW, aunque no es necesaria la construcción de un diccionario, ni de un *codebook*. Este método clasifica imágenes calculando la distancia de la imagen a clasificar a cada una de las clases del problema. La clase que se asigna a la imagen es aquella cuya distancia sea la menor de todas.

En definitiva este algoritmo se podría resumir en 5 sencillos pasos:

1. Se calculan todos los puntos característicos de la imagen Q a clasificar
2. Se calculan los puntos d_1……d_N siendo N el número de puntos característicos de la imagen Q a clasificar para todas las clases C, donde los puntos de las clases son asignados según la clase a la que pertenece la imagen.
3. Para todo punto d_i y para toda clase C, se calcula el vecino más próximo (NN) del punto d_i perteneciente a la clase C: $NN_c(d_i)$
4. Definimos la distancia de la imagen Q a la clase C como el promedio de la suma de distancias $\left(\frac{1}{N}\right)\sum_{i=1}^{n}\|d_i - NN_c(d_i)\|^2$.
5. Asignamos a la imagen Q la clase C más cercana.

$$C = \arg min_C \; 1/N \sum_{i=1}^{n} \|d_i - NN_c(d_i)\|^2$$

A pesar de la simplicidad del algoritmo, NBNN proporciona resultados satisfactorios en varias bases de datos [6; 38].

2.4 Evaluación de los resultados en la literatura

Un modelo de clasificación es una función que permite decidir dentro de un conjunto de datos cuáles pertenecen a un grupo o a otro, donde la diferencia entre un grupo y la otra será determinada por un valor umbral.

Una vez que están clasificadas las imágenes se procede a representar los resultados de la clasificación, estos se pueden mostrar de diferentes maneras. Existe el Error Absoluto, en el que se representa exactamente el número de fallos que se obtienen tras clasificar. También está la llamada matriz de confusión o tabla de contingencia, en la que se representan según la clase que se vaya a clasificar el número de aciertos y de fallos y curvas Receiver Operating Characteritics (ROC) y curvas Precision/Recall, en las que se representan gráficamente las tasas de éxito y tasas de falsos aciertos en clasificadores.

El problema de representar los resultados mediante el Error Absoluto es que mide el error para las clases conjuntamente. Es decir, si alguna clase es menos abundante que las otras, como ocurre en la mayoría de los problemas, el error de clasificación puede parecer bajo aunque realmente no lo sea. Por ejemplo, sí suponemos un problema binario, dos clases A y B, si la probabilidad a priori, (el número de objetos presentes antes de predecir), de la clase A es del 10% y de la clase B es del 90%, y decidimos predecir que todas las clases son del tipo B, mi error seria del 10%, pero si lo que decimos es lo contrario nuestro error sería de un 90%. Para detectar este problema existe lo que hemos definido anteriormente como matriz de confusión, pero con ello no resolvemos el problema del Error Absoluto, una de las posibles formas de resolver el problema del Error Absoluto es mediante las curvas ROC o Precision/Recall.

Si consideramos el problema de clasificación en el que se debe de predecir si en una imagen el objeto se encuentra presente (clase positiva) o no lo está (clase negativa), se tendría un problema de clasificación binaria. En este caso se tendrían cuatro posibles resultados. Si el resultado de una predicción es p y el valor real es también p, entonces se conoce como un Verdadero Positivo (VP), sin embargo si el valor real es n entonces se conoce como un Falso Positivo (FP). De la misma forma, tenemos un Verdadero Negativo (VN) cuando la predicción como el valor real es n, y un Falso Negativo (FN) si la predicción es n pero el valor real es p. Estos cuatro posibles resultados se suelen formular mediante una tabla de contingencia como la que se muestra en la Tabla 1, con la que podremos detectar el problema del Error Absoluto.

		Valor en la realidad		
		p	***n***	**total**
Predicción	***p'***	Verdaderos Positivos	Falsos Positivos	P'
	n'	Falsos Negativos	Verdaderos Negativos	N'
	total	P	N	

Tabla 1: Tabla de contingencia

Las curvas ROC (Receiver Operating Characteritics) son una técnica útil para analizar sistemas de detección y visualizar su comportamiento y poder así resolver el problema del Error Absoluto. Este tipo de gráficos se han utilizado en los últimos años en Teoría de Detección de Señales [44] para representar el equilibrio entre tasas de éxito y tasas de falsos aciertos en clasificadores.

Para analizar las curvas de Precision/Recall y las curvas ROC resultara útil definir algunos conceptos que se utilizan para la representación de estas curvas.

A continuación definimos algunos términos que hemos utilizado anteriormente y otros que vienen dados a partir de los anteriores.

Verdadero Positivo (VP): ejemplo positivo que se ha clasificado exitosamente.

Verdadero Negativo (VN): ejemplo negativo que se ha clasificado correctamente.

Falso Positivo (FP): ejemplo negativo que se ha clasificado como positivo.

Falso Negativo (FN): ejemplo positivo que se ha clasificado como negativo.

Razón de Verdaderos Positivos (VPR): razón de éxitos, también denominado sensibilidad, viene dado por la siguiente función:

$$VPR = \frac{VP}{VP+FN}$$

Razón de Falsos Positivos (FPR): razón de falsos éxitos, cuya expresión es:

$$FPR = \frac{FP}{FP + VN}$$

Recall = VPR.

Precision = VP/ (VP +FP).

2.4.1 El espacio ROC y valores de Precisión, Recall y AUC

Para dibujar una curva ROC solo serán necesarias las tasas de Verdaderos Positivos (VPR) y de Falsos Positivos (FPR).

La VPR mide hasta qué punto un clasificador es capaz de detectar los casos positivos correctamente, de entre todos los casos positivos disponibles durante la prueba.

La FPR describe cuantos resultados positivos son incorrectos de entre todos los casos negativos disponibles durante la prueba.

El espacio ROC se define por VPR y FPR como eje de ordenadas y abscisas respectivamente, y representa los intercambios entre verdaderos positivos y falsos positivos.

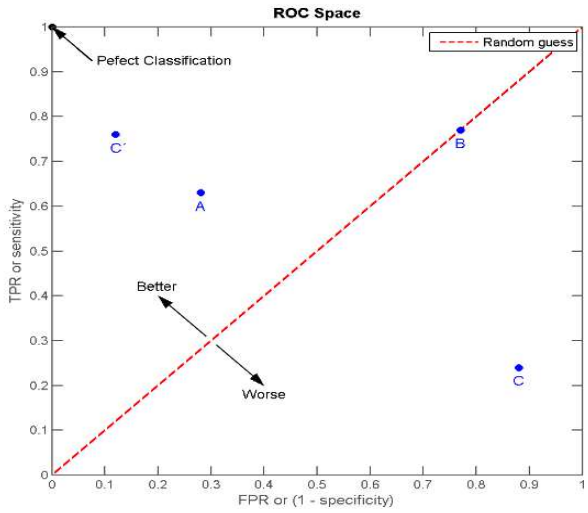


Ilustración 9: Ejemplos de Curvas ROC

El método con el mejor resultado posible se situaría en un punto en la esquina superior izquierda del espacio ROC, lo que se traduce en un clasificador que detecta el 100% de los casos positivos sin introducir ningún falso positivo, es decir, en una clasificación perfecta. Por el contrario, una clasificación totalmente aleatoria generaría una ROC a lo largo de la línea diagonal, que se llama también línea de no-discriminación, desde el extremo inferior izquierdo hasta la esquina superior derecha, como mostramos en la Ilustración 9 .

Para representar una curva de Precision-recall solo serán necesarios los valores definidos anteriormente como Precision y como recall, utilizando la tasa de Verdaderos positivos para recall y los valores de verdadero positivo y falso positivo para la precisión.

El espacio Precision-Recall se define por Precision en el eje de ordenadas y Recall en el eje de abscisas.

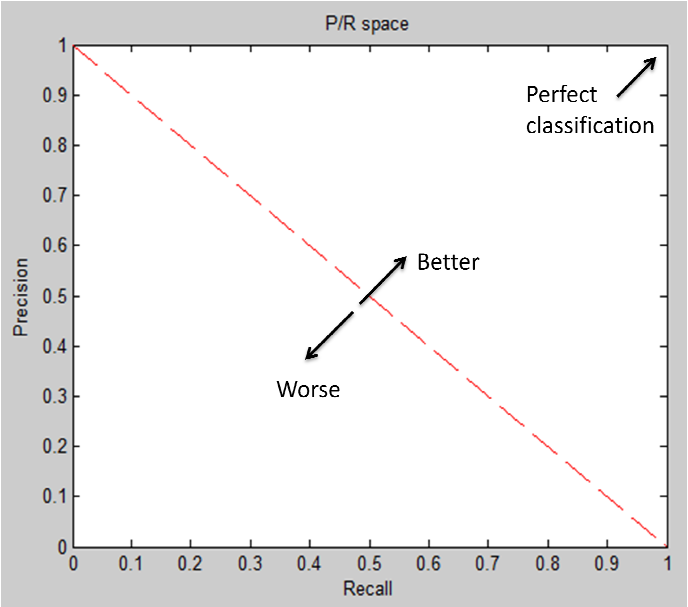


Ilustración 10: Curvas Precisión/Recall

El método con el mejor resultado posible se situaría en un punto en la esquina superior derecha del espacio P/R, como mostramos en la Ilustración 10.

Para resumir toda la información contenida en las curvas ROC y P/R utilizando un único valor numérico se suele utilizar el área bajo la curva o AUC. Nótese que en ambos casos el mejor resultado implica un AUC de 1:

Área Bajo la Curva (AUC), que es el valor del área que cubre la curva ROC o el Área que cubre la curva Precisión/Recall

En algunas bases de datos se recomienda el uso de medidas especificas, como es el caso de la Precisión media [5], que se define como la precisión medida en un rango de once divisiones equitativas de niveles de Recall y que introducimos como uno de los valores a utilizar ya que es la medida estándar para una de las bases de datos que vamos a utilizar.

$$AP = \frac{1}{11}\sum_{r\in\{0,0.1,\ldots,1\}} p_{interp}(r).$$

Donde la Precision de cada nivel de Recall *r* es el máximo valor medido entre dos valores consecutivos de *r*.

$$p_{interp}(r) = \max_{\hat{r}:\hat{r}\geq r} p(\hat{r})$$

2.5 Bases de Datos

En la literatura existen numerosas bases de datos de imágenes que se han utilizado como marco para evaluar y comparar algoritmos de categorización de imágenes, bases de datos públicas como Caltech-101 [45] o Caltech-256 [46], ambas bases tratan con imágenes de 101 clases diferentes o 256 clases, de cada clase existen aproximadamente unas 50 imágenes en cada base de datos.

A continuación se realizara con un estudio sobre dos de las bases de datos más relevantes, que además son las que se usarán en este proyecto. Pascal Challenge 2007 [5] y una base de datos descargada de [12] cuya referencia venia aportada por el articulo [13] donde se realiza un estudio con el SVM y a la que nombraremos como base de datos de Visual Geometry Group.

2.5.1 **Pascal Challenge 2007**

El desafío de Pascal es una competición anual que se lleva celebrando desde 2005 hasta la actualidad. En particular el desafío de Pascal de 2007 cuenta con 5011 imágenes etiquetadas y divididas en dos conjuntos (entrenamiento con 2501 imágenes y validación con 2510 imágenes), además se proporcionaba un conjunto de imágenes de test sin etiquetar. Todas las imágenes se obtuvieron automáticamente de Flickr [47] realizando búsquedas usando como palabra clave los nombres de las clases y las fotos eran posteriormente verificadas y etiquetadas por voluntarios. Se trata de una base de datos de 20 clases de imágenes entre las cuales hay coches, personas, aviones, botellas, vacas, perros, etc.

Los resultados son evaluados mediante curvas de *precisión/recall* y su correspondiente medida de la Precisión media (AP), todos estos valores han sido detallados en el apartado 2.4 de este proyecto. En la Tabla 2 mostramos los tres mejores algoritmos en el desafío Pascal 2007, cuyos métodos son versiones diferentes del BOW ('*bag-of-words*'). En la Ilustración 11 vemos un ejemplo de grafica *precisión/recall* de los 5 mejores algoritmos para la clasificación de coches utilizando como base de datos la de Pascal 2007.

	Bicicleta	Autobús	Coche	Gato	Caballo
INRIA_Genetic	0.636	0.606	0.78	0.588	0.775
INRIA_Flat	0.625	0.604	0.763	0.576	0.765
XRCE	0.575	0.575	0.754	0.503	0.757

Tabla 2: Precisión media según el modelo y la clase a clasificar para la base de datos de Pascal 2007

Como observamos en la Ilustración 11 la precisión media de aciertos del mejor algoritmo, en este caso INRIA_Genetic [48], es del 78% para la detección de coches, cuyo algoritmo de clasificación se base en SVM, al igual que los dos algoritmos siguientes INRIA_Flat [49] y XRCE [50].

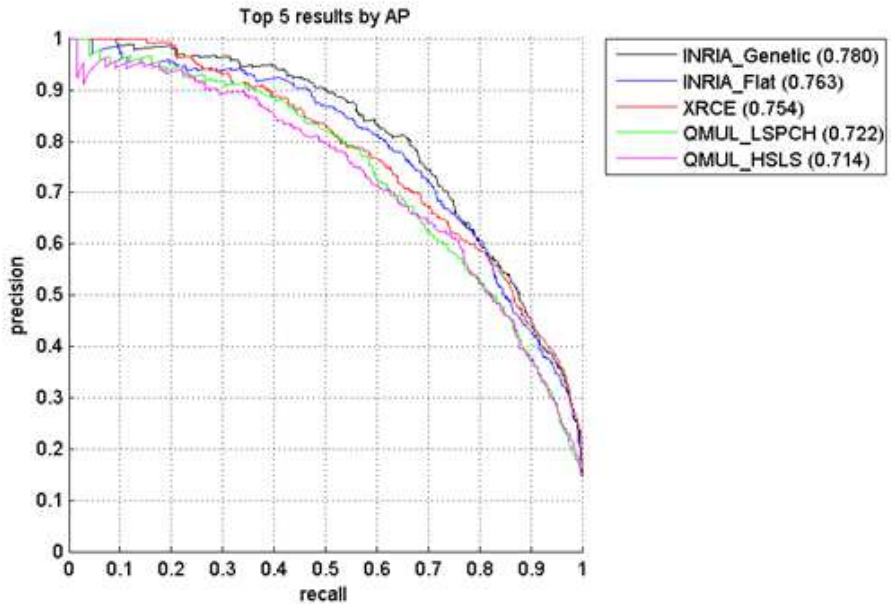


Ilustración 11: Grafica Precision/Recall para la detección de coches con los 5 mejores resultados en el Desafío Pascal 2007

Cada imagen viene dada con un archivo de anotaciones en el que se describe la imagen, tamaño, número de pixeles, el recuadro del/de los objetos presentes en la imagen, el número de objetos y la clase de los objetos.

Algunos ejemplos son mostrados en la Ilustración 12, en la que podemos observar 2 ejemplos de cada clase, (no están las 20 clases), el correspondiente recuadro en la imagen para la clase a la que se refiere. Las imágenes no son todas de las mismas dimensiones, y en una misma imagen puede encontrarse la presencia de varias clases como podemos observar en la Ilustración 13, donde también podemos comprobar que hay algunas clases cuya visión de ellas es bastante difícil, como en este caso la presencia de coches en la imagen.

Ilustración 12: Ejemplo de imágenes contenidas en la base de datos Pascal 2007.

En la Tabla 3 se muestra el número de imágenes y objetos de cada clase que se encuentra en la base de datos, ya que muchas imágenes contienen objetos de diferentes clases, el total en la columna no es el mismo que la suma de sus columnas correspondientes.

Ilustración 13: Ejemplo de imagen que contiene varias clases en la misma foto

	train		val	
	images	objects	images	objects
Aeroplane	112	151	126	155
Bicycle	116	176	127	177
Bird	180	243	150	243
Boat	81	140	100	150
Bottle	139	253	105	252
Bus	97	115	89	114
Car	376	625	337	625
Cat	163	186	174	190
Chair	224	400	221	398
Cow	69	136	72	123
Dining table	97	103	103	112
Dog	203	253	218	257
Horse	139	182	148	180
Motorbike	120	167	125	172
Person	1,025	2,358	983	2,332
Potted plant	133	248	112	266
Sheep	48	130	48	127
Sofa	111	124	118	124
Train	127	145	134	152
Tv/monitor	128	166	128	158
Total	2,501	6,301	2,510	6,307

Tabla 3: Número de imágenes y objetos por clase

2.5.2 Base de datos de Visual Geometry Group

En esta base de datos solamente hay 5 clases que distinguir, como son caras (450 imágenes), coches vistos desde atrás (1155 imágenes), coches vistos de lado (720 imágenes), aviones (1074 imágenes) y motos (826 imágenes). En este caso las imágenes no tienen ninguna anotación y tampoco están separadas en entrenamiento y en test.

Esta base de datos es bastante más sencilla que la del desafío de Pascal, como podemos observar en la Ilustración 14, en la que las imágenes de los objetos están centradas sólo en ese objeto y no incluyen otros objetos dentro de la misma imagen, por lo que la dificultad para clasificar las imágenes se reduce bastante. En la Tabla 4 podemos observar la matriz de confusión para el modelo BOW+SVM con k=1000 clusters. Ayudándonos de los valores de esta tabla y la cantidad de imágenes que hay en la base de datos podemos calcular el valor de P/R, por ejemplo, si calculamos el valor de precisión para el coche visto desde atrás nos da un valor de 98% para un recall del 97.7%, en cambio si observamos la Ilustración 11 para ese mismo recall se observa que la precisión no supera valores del 30 % para ninguno de los 5 mejores algoritmos.

Un problema de esta base de datos es que el enlace de descarga para las imágenes de coches vistos desde el lado no está operativo por lo que no se pueden descargar, por lo tanto nuestras pruebas serán realizadas sin este conjunto de imágenes, es decir solo utilizaremos 4 clases.

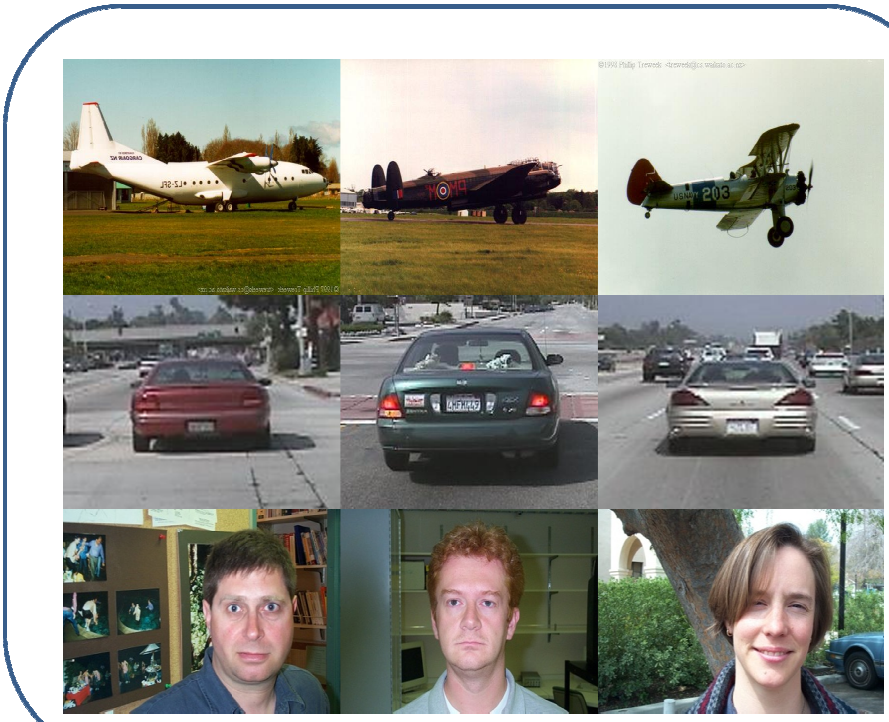


Ilustración 14: Ejemplos de imágenes en la base de datos

True classes →	*faces (frontal)*	*airplanes (side)*	*cars (rear)*	*cars (side)*	*motorbikes (side)*
faces(frontal)	**94**	0.4	0.7	0	1.4
airplanes (side)	1.5	**96.3**	0.2	0.1	2.7
cars (rear)	1.9	0.5	**97.7**	0	0.9
cars(side)	1.7	1.9	0.5	**99.6**	2.3
motorbikes (side)	0.9	0.9	0.9	0.3	**92.7**
Mean ranks	1.07	1.04	1.03	1.01	1.09

Tabla 4: Matriz de confusión para SVM

3. Diseño e implementación

En el capítulo anterior se abordaban los sistemas más relevantes encontrados en la literatura, en este capítulo se explica las implementaciones realizadas basadas en dichos sistemas. La herramienta de software que utilizaremos fundamentalmente en la implementación será Matlab, aunque también utilizaremos funciones en C++ para crear ejecutables para Matlab.

En general, nuestra implementación se puede resumir en una serie de pasos que hemos ido siguiendo para poder obtener resultados satisfactorios en la detección de objetos en las imágenes. En la Ilustración 15 observamos el camino seguido para la elaboración del proyecto.

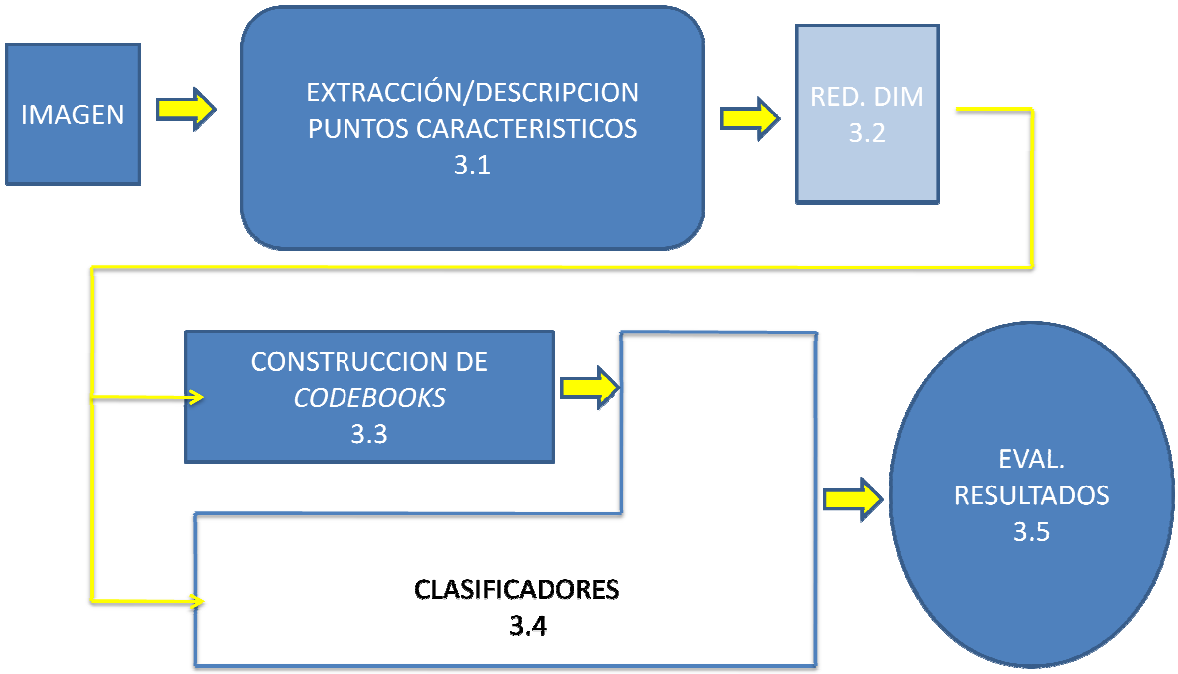


Ilustración 15: Gráfico que ilustra los pasos seguidos para la implementación y diseño de este proyecto

Existen 5 módulos principales en el diseño de nuestro proyecto:

3.1.- Extracción/descripción de puntos característicos: En este modulo se detectarán los puntos característicos utilizando dos métodos diferentes, uno de ellos se le aplica el algoritmo SIFT directamente y en el otro método se escogen los puntos utilizando posiciones espaciales predeterminadas y se aplicara SIFT sobre esos puntos para que nos devuelva el descriptor.

3.2.-Reduccion de la dimensionalidad: Utilizando PCA se reducirá la dimensión de los vectores de características extraídos por SIFT.

3.3.-Construcción de *codebook:* Una vez extraídos los puntos característicos, habrá que caracterizar a las imágenes con el mismo número de características, para ello se utilizara el algoritmo *k-means.*

3.4.-Clasificación: Como último paso de nuestro diseño habrá que clasificar las imágenes, para ello utilizaremos dos métodos diferentes, uno SVM para el cual necesitamos los *codebook* y NBNN para el cual no necesitamos los *codebook*.

3.5.-Evaluación de resultados: Finalmente se evaluarán estas clasificaciones mediante una serie de curvas como son las Curvas ROC y Precisión/Recall y unos valores numéricos para resumir estas curvas como la Precisión media o el Área bajo la curva.

3.1 Extracción de características

Según hemos visto en el capítulo *Estado del Arte*, una de las etapas fundamentales en la categorización de imágenes es la extracción de características de las imágenes tanto a analizar como a clasificar, ya sean características de color, detectores de esquinas, bordes o puntos característicos o combinaciones de ellos.

Y uno de los métodos más utilizados para la extracción de estas características es el algoritmo SIFT, ya sea aplicando a la imagen directamente este algoritmo y obtener los puntos característicos correspondientes, o eligiendo puntos de la imagen y obtener sus descriptores mediante SIFT.

El segundo método comentado anteriormente de elegir los puntos y describirlos es uno de los métodos con mejores resultados del *Desafío Pascal 2007* utilizado por [7].

3.1.1 Extracción de puntos SIFT

El algoritmo SIFT usado en nuestra implementación [31], se usa para la extracción de los puntos más característicos de las imágenes. Este algoritmo es muy utilizado en diversos campos de detección y reconocimiento tanto de objetos como de humanos [15] .Esta implementación del SIFT toma una imagen y la transforma en un número de vectores de características locales. Cada uno de estos vectores, formados por 128 descriptores, es invariante a escala, rotación o traslación de la imagen.

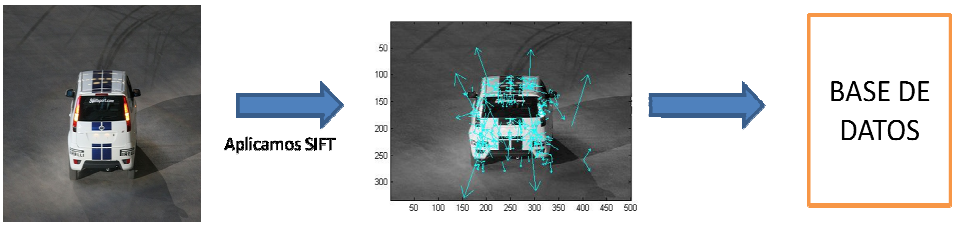


Ilustración 16: Aplicación de SIFT a las imágenes

Una vez realizada la base de datos con todos los puntos de las imágenes de entrenamiento aplicaremos los respectivos algoritmos para clasificar las imágenes de test. A las imágenes de test también tendremos que extraerles sus puntos característicos y así poder clasificarlos según el algoritmo correspondiente.

Este tipo de extracción de puntos característicos causa algunos problemas, uno de estos problemas es que no todas las imágenes tienen el mismo número de puntos de interés, el número de puntos depende de las características de la imagen (resolución, contraste, etc.). Otro problema es que el algoritmo utilizado para la extracción de los puntos (SIFT) nos devuelve los descriptores normalizados, por lo que el cálculo de las distancias no se podrá realizar mediante distancia Euclidea.

3.1.2 Extracción de características sobre un espacio predeterminado

En el apartado 3.2 explicamos cómo extraemos los puntos característicos mediante la aplicación del algoritmo SIFT [31], en este apartado se describirá otro método de extracción de puntos de interés, pero en este caso la extracción será diferente. Los puntos serán escogidos de una manera uniforme sobre toda la imagen utilizando una cuadricula que determinaremos nosotros y posteriormente aplicaremos el algoritmo SIFT [30] para que nos devuelva los descriptores de cada punto. Esta forma de describir las imágenes nos da una serie de ventajas, como que todas las imágenes tengan el mismo número de puntos, y se caracterizan casi todas las zonas de la imagen.

El problema es que los puntos no son invariantes a la orientación y ni a la localización.

A todas las imágenes, tanto imágenes de entrenamiento como de test, se les aplicará el mismo método, que se basa en los siguientes puntos como mostramos en la Ilustración 17:

- Cambio de escala de todas las imágenes
- Mallado de las imágenes
- Aplicación de región circular sobre cada punto del mallado con un total de 5 escalas diferentes sobre cada punto
- Caracterización de las regiones circulares usando el descriptor SIFT

Todo este proceso descrito, se realiza para poder obtener un número de puntos característicos iguales en todas las imágenes. El cambio de escala en las imágenes lo realizamos para poder tener en todas las imágenes el mismo número de pixeles, el mallado consistirá en el cuadriculado de la imagen quedándonos con el punto de cada intersección, para realizar el mallado la imagen será dividida en tantos puntos como características queramos obtener, se realizaran unas regiones circulares sobre esos puntos y abarcando lo máximo posible de la imagen, y finalmente la obtención de los descriptores de cada punto en esas regiones circulares con los que crearemos nuestra base de datos. Realizaremos diferentes divisiones para obtener diferentes números de puntos, sacando 500, 1000, 2000 y 5000 puntos característicos de cada imagen. En la Ilustración 17 observamos una explicación grafica de este método.

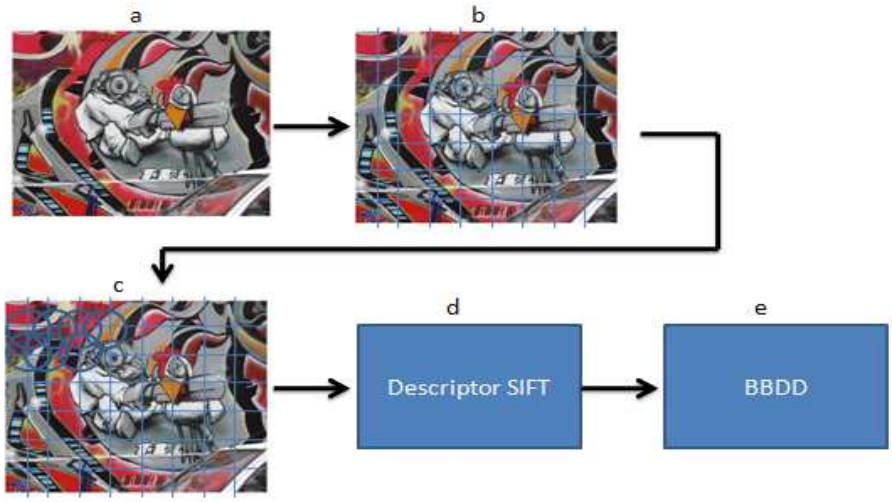


Ilustración 17: Proceso para la extracción de puntos característicos, a) imagen escalada, b) imagen mallada, c) Regiones circulares en la imagen mallada, d) descriptor SIFT y e) Base de datos

3.2 Reducción de la dimensión

Una vez extraídas las características de las imágenes, están representadas por vectores de alta dimensionalidad (SIFT 128 componentes). Estos vectores pueden contener información poco relevante además de que, al ser elementos tan grandes, el coste computacional es mayor causando así un funcionamiento más lento de los sistemas que los usen, por lo que se harán algunas pruebas reduciendo la dimensión de los vectores para comprobar si se mejoran los resultados.
El método utilizado para la reducción de dimensionalidad es PCA (Manual del programador A.A), quedándonos con 100 componentes de las 128 que nos devuelve SIFT.

3.3 Construcción del codebook

Una vez terminada la base de datos con todos los puntos característicos de las imágenes de entrenamiento (por los dos métodos explicados), se procederá a la creación de un diccionario de palabras visuales.

Se realizarán dos tipos de diccionarios, uno en el que se utilizarán todas las imágenes de entrenamiento para crear los *clusters* y otro en el que se separarán las imágenes de entrenamiento en dos partes, un 60 % para realizar los *clusters* y el 40% restante para entrenar el clasificador. Del 60% de las imágenes escogidas se igualaran clases, es decir, igualaremos el número de imágenes que contienen el objeto deseado con el número de imágenes de clase contraria. Realizaremos diferentes pruebas variando el valor de k, los valores utilizados son 100, 200,300, 500, 700, 1000. Estos valores de k los utilizaremos para el método de extracción de puntos mediante SIFT. Se han realizado pruebas adicionales recortando las imágenes y quedándonos sólo con los puntos característicos de los objetos y obviando los del fondo (ver capitulo 3.4.1). Al recortar las imágenes el número de puntos decrece por lo que también tendremos que disminuir los valores de k, utilizaremos 50, 80, 100, 150, 200, 250, 500 y 700 *clusters*.

La implementación del algoritmo *k-means* utilizada es la proporcionada por matlab aunque con algunos parámetros y modificaciones introducidas, ya que por problemas de memoria de matlab y para que sea más rápido hemos tenido que realizar cambios (Manual del programador A.A). Una de las opciones introducidas será la forma en que se le introducen los datos, no se introducirán todos los puntos a la vez, sino que se le introducirán un número determinado de puntos y luego al resto se les asignara un *cluster* dependiendo de la cercanía a la que este, otra modificación introducida será la opción de evitar que busque el *cluster* más eficiente, para evitar que el algoritmo dure mucho tiempo. Para evitar posibles imperfecciones en la elección de los *clusters*, este proceso de dividir las imágenes y la realización de los *clusters*, se repetirá 10 veces y posteriormente se hará una media de los resultados.

Se han realizado también pruebas con una versión diferente de k-means descargado de la pagina web [51], que utiliza un procedimiento para la elección de los centros de los *clusters* llamado *EZ_Hybrid* [22]. Este método se basa en un simple algoritmo hibrido, en el que se realiza primeramente, una búsqueda local heurística en el que se van intercambiando los centros entre unos que existen y otros que son candidatos a ellos [35], y posteriormente se aplica un número de iteraciones con el algoritmo de Lloyd [22].

3.4 Algoritmos de clasificación

Una vez caracterizadas todas las imágenes con el mismo número de características, tanto las imágenes de entrenamiento como las imágenes de test. Es el momento de clasificar las imágenes por su contenido. Para ello vamos a utilizar dos métodos de clasificación, NBNN y SVM. Las

imágenes de entrenamiento en algunos casos también han sido divididas en dos grupos un 60% para la creación de los *clusters* y un 40% para entrenar el clasificador.

3.4.1 Naive-Bayes Nearest Neighbor (NBNN)

El algoritmo de clasificación NBNN [6], a diferencia de otros no necesita entrenar un clasificador, directamente asigna la imagen de test a la clase cuya distancia sea más cercana, por lo que no necesita del paso previo de la creación de *codebook*.

Hemos aplicado este clasificador de dos maneras diferentes, aplicando un recorte a las imágenes de entrenamiento y quedándonos solo con el objeto u objetos que se encuentran en ella y así solo se tendrá en cuenta los puntos característicos que se encuentren dentro de esos objetos, o sin aplicar el recorte y quedándonos con todos los puntos característicos de la imagen.

- SIN RECORTE: los puntos característicos guardados en la base de datos estarán clasificados mediante la etiqueta de la clase a la que pertenecen. Sí la imagen a la que pertenece el punto contiene el objeto a buscar estará clasificada como clase positiva, por lo tanto todos los puntos de esa imagen tendrán clase positiva, y si no lo contiene será de clase negativa. Una vez realizada esta división de clases, obtendremos los puntos característicos de las imágenes de test y una a una se irán clasificando.
  El método que se seguirá para clasificar será mediante el vecino más cercano, esto se realizará mediante el cálculo de la distancia entre un punto de la imagen de test y todos los puntos de las imágenes de entrenamiento, quedándonos con la mínima distancia y su clase. A continuación, se calculará un promedio de las mínimas distancias tanto para la clase negativa como para la clase positiva con respecto al total de los puntos característicos que haya en la imagen de test. Obteniendo al final para cada imagen de test las distancias a la clase negativa y a la clase positiva (ver capitulo 2.3).
  $distmin = {}^{1}/_{N} \times \sum_{i=1}^{N} dist_{min(i)}C$, $siendo\ N = n^{\underline{o}}\ puntos\ de\ la\ imagen\ de\ Test$ y $dist_{min(i)}C$ la distancia del punto i al más cercano de la clase C.

  Una vez calculada la distancia mínima de una imagen de test a cada una de las clases existentes en la base de datos, se decide la clase a la que pertenece comprobando cual es la distancia mínima.
  Se realizaran tres tipos de pruebas:
  - Una en el que el número de imágenes de ambas clases sea el mismo.
  - Otra en el que el número de imágenes de clase negativa sea mayor (60%) que el de la clase positiva (40%).
  - Otra en el que el número de imágenes de clase positiva (60%) sea mayor que el de clase negativa (40%).

  Estas pruebas las realizaremos para comprobar si nuestros resultados dependen del número de imágenes que exista en nuestra base de datos de cada clase.
  Finalizadas estas pruebas se procede al cálculo de las curvas ROC y P/R, para ello ordenamos todas las imágenes de test según la distancia calculada en orden ascendente, esta distancia es la relación entre la diferencia de las distancias de la clase positiva y de la clase negativa y la suma de la clase positiva con la clase negativa.

$$distnueva = {(distmas - distmenos)}/{(distmas + distmenos)}$$

Donde distmas es la distancia promedio de una imagen de test a la clase positiva y distmenos es la distancia promedio a la clase negativa.
Cuanta más pequeña sea esta diferencia de distancias más probable que sea de clase positiva y cuanto más grande más probable de clase negativa.

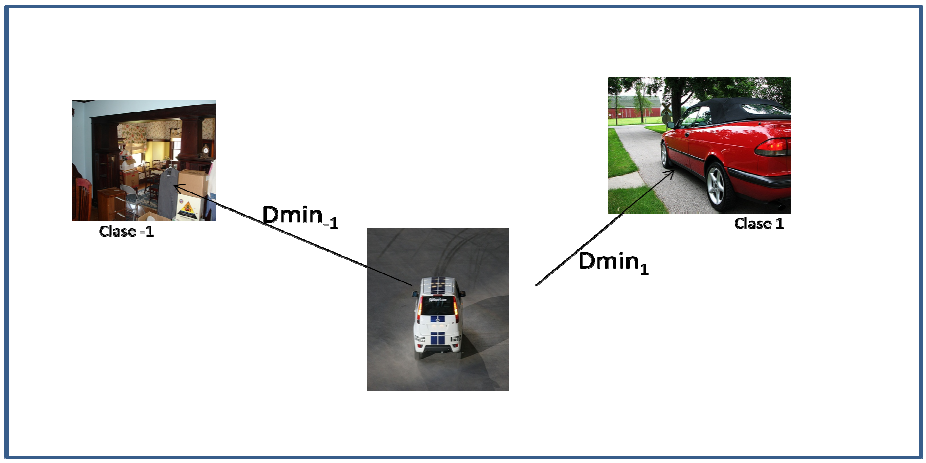


Ilustración 18: Distancias entre la imagen de Test (parte inferior) y las de Entrenamiento (parte superior)

- CON RECORTE: En este caso se recortaran todas las imágenes de entrenamiento, quedándonos con los objetos más destacados de la imagen y por lo tanto con sus correspondientes puntos característicos. Se descartará el fondo de todas las imágenes. Todas las imágenes, tanto de entrenamiento como de test, serán recortadas, eliminando el fondo, la base de datos estará construida únicamente por los objetos que se quieren detectar, es decir, en el Desafío Pascal 2007 se clasifican 20 objetos, estos son los que formarán la base de datos.

  En la Ilustración 19, observamos cómo se aplica el método de recorte, en un principio sacamos de la imagen todos los puntos característicos obtenido mediante el detector SIFT, en realidad todos estos puntos ya habían sido almacenados en una matriz, simplemente ahora lo que se realizará será una separación tanto de las clases como del fondo, esto se hará mediante la comprobación de si el punto está dentro de un límite de la caja recuadrada en rojo o no, estos márgenes nos vienen dados por la base de datos construida por VOC PASCAL 2007, naturalmente habrá puntos que se confundan entre las dos clases como vemos en la figura ya que en este caso el recuadro limitado por el caballo también engloba al del hombre en su mayor parte, y viceversa, el del hombre contendrá puntos pertenecientes al caballo, pero aún así habremos conseguido separar algunos puntos de otros.

Estas pruebas serán realizadas únicamente con las imágenes de Pascal 2007 ya que también incluyen una serie de anotaciones como son en que pixeles se encuentra el objeto.

Una vez visto los resultados (ver capitulo 4.4.1) mediante el recorte y la utilización de NBNN como método de clasificación, nos decantaremos por este algoritmo como clasificador del sistema final.

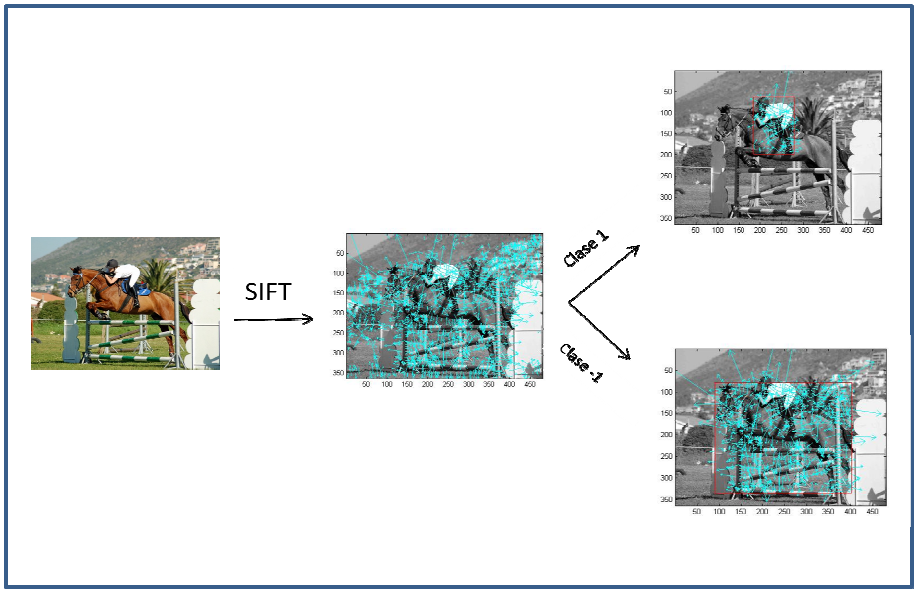


Ilustración 19: Recorte de imagen de entrenamiento

3.4.2 Support Vector Machines (SVM)

Una vez construidos los diccionarios visuales (*codebooks*) pasamos a entrenar el clasificador SVM. Como hemos explicado anteriormente, tenemos dos tipos de de diccionarios visuales y cada uno de ellos puede tener distintos ks, uno que está formado por todos los puntos de todas las imágenes de entrenamiento y otro que estará formado por el 60 % de las imágenes de entrenamiento en el que a su vez se igualarán clases.

El proceso de entrenamiento del clasificador SVM seguirá la misma metodología que con NBNN, primeramente se utilizarán imágenes sin recortar y posteriormente recortadas. El clasificador SVM utilizado en nuestro caso es el que nos proporciona la librería de matlab libsvm [52]. Esta librería nos proporciona dos funciones parecidas a las que tiene matlab.

Svmtrain y Svmpredict, la primera de ellas como su nombre nos da a entender sirve para entrenar el clasificador, a esta función se le introducirán los vectores descriptores de cada imagen de acuerdo al diccionario visual, con sus correspondientes etiquetas y una serie de opciones, como son el *kernel* que utiliza, que en nuestro caso y que en algunos artículos como [13] y [14], elegiremos un *kernel* lineal, otra opción será el parámetro de complejidad C de la función de coste visto en 2.2.4 que le asignaremos un valor pequeño fijado en 0.005, esta función nos devolverá un modelo del entrenamiento que utilizaremos para predecir con las imágenes de test de qué clase son.

En el caso de usar las imágenes en las que utilizamos como puntos de interés los devueltos por el algoritmo SIFT [18] directamente, al observar los resultados decidimos realizar una proyección

Fisher [53] de los resultados obtenidos por SVM para comprobar la manera en que son separados los puntos y si están diferenciados o no.

3.5 Evaluación de los resultados

Para la evaluación de los resultados, utilizamos gráficas Precisión/Recall y curvas ROC. También utilizaremos valores numéricos, como el valor del Área bajo la curva (AUC) ROC y la Precisión media de la grafica de Precisión/Recall.

Estos resultados se obtienen tras la utilización de los clasificadores SVM y NBNN. Para conseguir los resultados con SVM, se realiza una ordenación de las imágenes usando una estimación de la probabilidad de pertenencia de la imagen de test a las clases, Y para NBNN en vez de utilizar una estimación de la probabilidad, la ordenación de las imágenes se realizará mediante una clasificación por distancia mínima de la imagen de test a las clases, como hemos explicado en el capitulo 3.4.1.

3.6 Bases de datos utilizadas

3.6.1 Pascal 2007

La base de datos descargada de Pascal 2007 la utilizaremos para separar/distinguir imágenes que contengan coches del resto de las imágenes que no contengan coches. Utilizaremos como imágenes de entrenamiento las imágenes proporcionadas por este desafío, y como imágenes de test las ofrecidas como imágenes de validación.

En esta base de datos hay un total de 5011 imágenes, de las cuales 2501 son usadas como imágenes de entrenamiento, y 2510 como imágenes de test.

En total hay aproximadamente 8000 objetos tanto en entrenamiento como en test, como objetos definimos a tanto personas, coches, animales, botellas (las 20 clases definidas en Pascal 2007), de las cuales solo un 10% son coches.

Para observar la dificultad del problema lo primero que se hizo fue comprobar mediante la proyección Fisher, lo bien que está separando los puntos característicos después de haber utilizado *k-means*.

El análisis discriminante de Fisher (FDA) propone como solución maximizar una función que represente la diferencia entre las medias, normalizada por una medida de la dispersión dentro de las clases. El discriminante lineal de Fisher se define como la función lineal w^Tx que maximiza la función objetivo:

$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{S}_1^2 + \tilde{S}_2^2}$, este discriminante nos produce:

- La dispersión entre-clases en la proyección disminuye
- La diferencia entre medias proyectadas aumenta

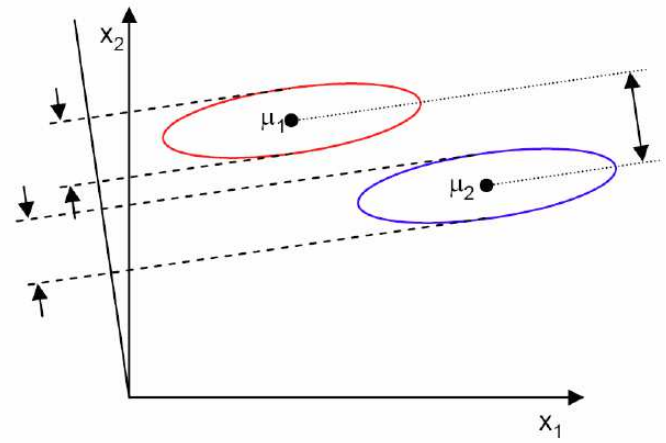
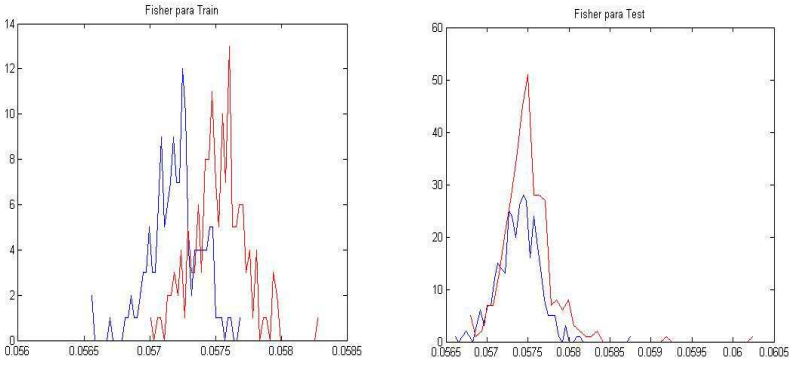


Ilustración 20: Una proyección donde los ejemplos de la misma clase son proyectados muy cerca unos de otros, y al mismo tiempo, las medias proyectadas están lo más lejos posible.

En nuestro caso, la proyección Fisher nos proporciona los siguientes resultados, observamos que a medida que aumentamos el número de clusters más separadas están las medias, en el caso de las imágenes sin recortar a medida que aumenta el número de clusters mejora la proyección. En estos casos que se muestran, siendo k el número de *clusters* que se ha utilizado en *k-means*, como observamos en la parte de entrenamiento esta separación de clases es clara, pero en cambio cuando lo aplicamos a los puntos de test ya no es tan clara, por lo que se puede decidir que son difíciles de clasificar. El color de cada curva representa una clase diferente.

K=300:



K=700:

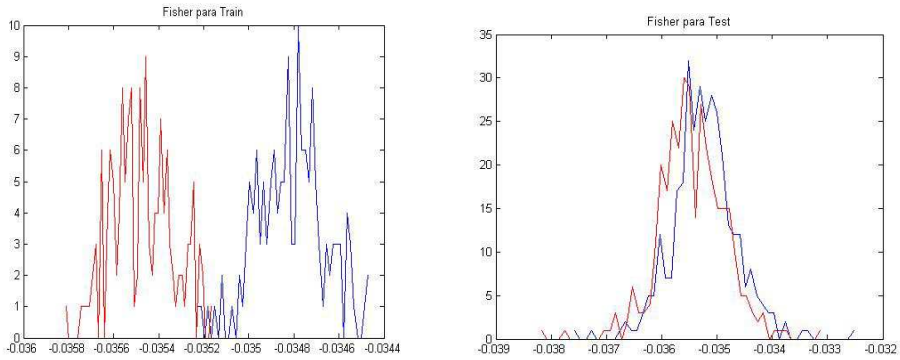


Ilustración 21: Proyecciones Fisher para diferentes números de clusters para imágenes sin recortar

Vistos los resultados con los puntos característicos extraídos directamente con SIFT [31], se decidió seguir un procedimiento de obtención de puntos aplicando una cuadrícula a la imagen y posteriormente aplicar el descriptor SIFT para la obtención de los puntos característicos.

3.6.2 Base de datos Visual Geometry Group

Esta base de datos será dividida manualmente, un grupo de imágenes para entrenamiento y otro para test. En total hay 3505 imágenes, de las cuales 1155 son coches y el resto se reparte entre aviones, personas y motocicletas.

Nos vamos a centrar en la clasificación de coches vistos desde atrás, y no utilizaremos los coches vistos desde el lado ya que como dijimos anteriormente la pagina de descarga esta inutilizada.

Para la división de imágenes de entrenamiento y test, igualaremos clases en ambos casos como vemos en la Tabla 5.

	Entrenamiento	Test
Coches	555	600
Resto de objetos	555	600

Tabla 5: Número de imágenes que contienen entrenamiento y test para la base de datos de Visual Geometry Group

3.7 Base de Datos Privada

En este caso, también haremos una serie de pruebas con una base de datos privada. Estas imágenes que nos proporcionan, son todas ellas obtenidas, desde el mismo punto de vista, como podemos observar en la Ilustración 22.

Ilustración 22: Ejemplos de la Base de datos privada

Todas las imágenes proporcionadas vienen clasificadas según el tipo de vehículo, y solo se tienen ejemplos de autobuses y coches, por lo que se intentará diferenciar coches de autobuses dentro

de un entorno controlado, como es el caso de que sólo aparezcan en las imágenes uno de estos dos tipos de vehículos (coche o autobús). Hay un total de 447 autobuses y de 325 coches.

También se incluyen imágenes de los vehículos en sí, es decir, el vehículo recortado como podemos observar en la Ilustración 23. En imágenes recortadas tenemos 12 coches y 10 autobuses. Para entrenar las imágenes se utilizaran todas las imágenes menos una que se utilizará para test.

Ilustración 23: Ejemplos de imágenes recortadas

4. Integración, pruebas y resultados

En el desarrollo de este capítulo se van a presentar diferentes resultados experimentales obtenidos para los sistemas implementados. Estos sistemas son implementados utilizando distintas técnicas ya descritas en el capítulo anterior.

Se analizarán los resultados obtenidos mediante curvas ROC y curvas de precisión/recall y los correspondientes valores del área bajo la curva de la ROC y de la Precision Media (AP) de los cuales ya hablamos en el capítulo 2.4. A partir de ellos compararemos las diferentes propuestas y se extraerán las conclusiones de las diferentes soluciones en las que se ha trabajado, para poder así diseñar el sistema final.

4.1 BOW+SVM con extracción de características SIFT en Base de datos Geometry Group

En este apartado expondremos los resultados obtenidos mediante la aplicación del método de *'Bag-of-Words'* con SVM como clasificador en una base de datos descargada del artículo [13].

En este caso, para obtener estos resultados, se extraen los puntos característicos mediante el algoritmo SIFT [31] , posteriormente se crea un diccionario visual con el algoritmo *k-means* utilizando 1000 clusters, en este caso la parte de entrenamiento está separada en un 60% para realizar el *k-means* y un 40% para entrenar el SVM, como ya describimos en el capítulo 3.3, y ya por último se aplica el clasificador SVM, con sus correspondientes parámetros como son con una función lineal para el *Kernel* y un coste computacional de C=0.005 como explicamos en el capitulo 3.4.2, para seleccionar las imágenes que contengan el objeto deseado. En nuestro caso, el objeto a buscar es un coche, cuya vista será desde atrás, como podemos ver en la Ilustración 24.

Ilustración 24: Imagen de ejemplo de un coche

La Ilustración 25 muestra las curvas P/R y la curva ROC para la clasificación de coches y la Tabla 6 muestra los valores de AUC para la curva ROC y la AP de la curva P/R. Son resultados para unas imágenes de Test y de Entrenamiento que hemos seleccionado nosotros, por lo que los resultados no pueden compararse exactamente al artículo [13], pero aun así podemos compararlo con el valor de Precision (98%) que obtuvimos en el capitulo 2.5.2 para un Recall del 97%. En nuestro caso la Precisión que tenemos es de un 70%, esta diferencia puede ser debida a la forma de dividir las imágenes de entrenamiento y de test que no explican en el articulo [13] o a la forma de construir el diccionario, a posibles diferencias en el preprocesado de las imágenes o a la extracción de características.

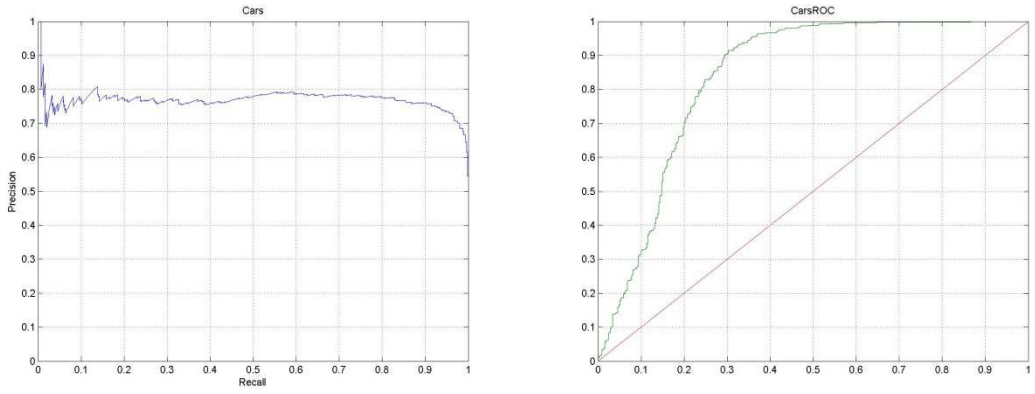


Ilustración 25: Gráfica Precision Recall y ROC para coches en base de datos Visual Geometry Group.

BOW (k)	AP (%)	Priori (%)	AUC (%)
1000	78.50	55	83.89

Tabla 6: Valores de Precision media (AP) y el Área bajo la Curva (AUC) para la base de datos Visual Geometry Group. Imágenes ordenadas en el CD adjunto.

4.2 Aplicación de BOW+SVM en Pascal 2007

Aplicaremos el mismo método que utilizamos con la base de datos VIsual Geometry Group y que hemos visto en el anterior apartado, a una base de datos más complicada (ya que las imágenes en este caso no son solo del objeto a encontrar, ni tampoco se encuentra en el centro de la imagen, como vemos en los ejemplos mostrados en la Ilustración 12).

Utilizando el mismo método ya explicado en el anterior apartado, extracción de características mediante SIFT [31], creación de un diccionario visual con *k-means* con k=1000 clusters y aplicando SVM como clasificador, los resultados obtenidos son los que observamos en la Tabla 7 y en la Ilustración 26.

BOW (k)	AP (%)	Priori (%)	AUC (%)
1000	28.17	14	71

Tabla 7: Valores de Precision media (AP) y el Área bajo la Curva (AUC) para Pascal 2007. Imágenes ordenadas en el CD adjunto.

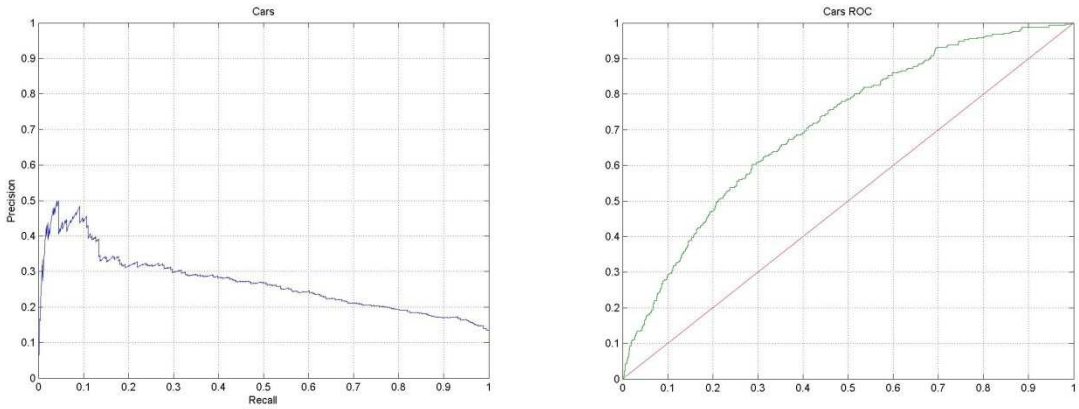


Ilustración 26: Resultados obtenidos mediante BOW+SVM con k=1000. Gráfica izqda. curva Precision Recall y grafica de la dcha. curva ROC.

Vistos los resultados obtenidos y comprobando que no son del todo satisfactorios en comparación con los mostrados en el desafío Pascal 2007 y que mostramos en el capitulo 2.5.1, decidimos estudiar diferentes alternativas para intentar mejorar estos resultados y que se nos adapten más a la literatura estudiada cuyos métodos son muy similares al descrito. Para intentar replicar estos resultados iremos aplicando varias ideas para mejorar los resultados. Existen varias formas de mejorarlos, uno de ellos será variar el número de *clusters*, otro método será recortar las imágenes quedándonos con el objeto u objetos en cuestión para la clasificación de las imágenes y por último variaremos la forma de extraer los puntos de interés de las imágenes como hacen en [7].

4.2.1 Variación del número de clusters

En este apartado estudiaremos la posibilidad de mejorar los resultados obtenidos en el apartado 4.2, variando el número de palabras visuales, es decir, el número de clusters. Se mantiene el mismo método aplicado en los dos anteriores apartados salvo que en este caso el número de *clusters* varía, utilizando valores de k=100, 200, 300, 500, 700, y 1000 *clusters*.

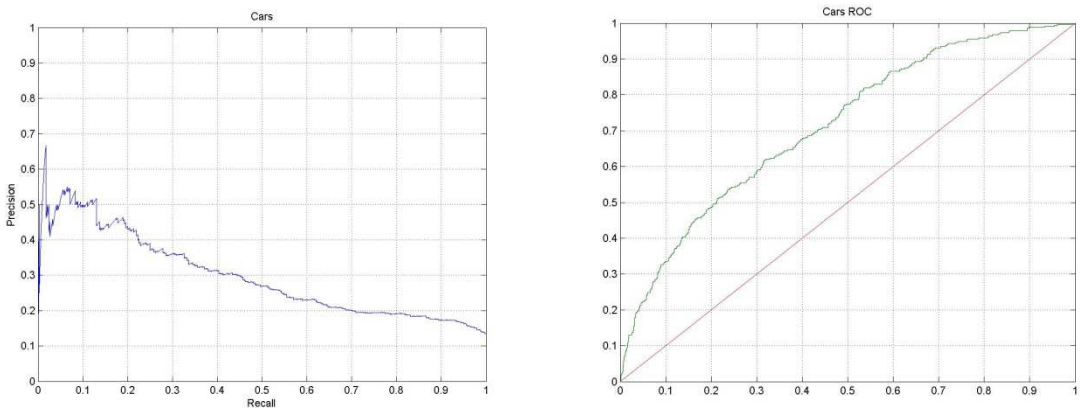


Ilustración 27: Gráficas Precision Recall (izqda.) y ROC (dcha.) para k=700.

Como observamos en la Tabla 8, a medida que aumentamos el número de clusters, tanto la precisión media como el área bajo la curva ROC mejora, aunque como hemos visto hasta un límite ya que para k=1000 clusters vuelve a empeorar.

BOW (k)	AP (%)	Priori(%)	AUC (%)
100	20.91	14	64.52
200	24.55	14	67.04
300	25.61	14	66.75
500	24.75	14	67.93
700	**31.82**	14	**71.38**
1000	28.17	14	71

Tabla 8: Valores de AUC y AP para diferentes números de clusters, los valores resaltados son aquellos con mayor acierto. Imágenes ordenadas en el CD adjunto.

4.2.2 Recortando las imágenes (Bounding Box)

Otra alternativa que proponemos y aprovechando que las imágenes descargadas de la base de datos de Pascal 2007 [5], vienen acompañadas de una serie de anotaciones, como por ejemplo de los objetos que hay en la imagen y la situación de estos en ella, es recortar las imágenes quedándonos con los objetos de interés. Es decir, separaremos los objetos del fondo y dividiremos los objetos en dos clases, una clase con los objetos que hay que detectar (clase positiva) y otra clase con los objetos que no hay que detectar (clase negativa). Al recortar las imágenes la parte de entrenamiento se nos queda en 7817 imágenes (826 de clase positiva y 6991 de clase negativa) y la parte de test en 7785 imágenes (817 de clase positiva y 6968 de clase negativa).

A estas nuevas imágenes se les aplicara el mismo método que en los apartados descritos anteriormente, salvo que introduciremos otros valores de k, ya que como es obvio el número de puntos característicos ha disminuido considerablemente.

BOW (k)	AP (%)	Priori (%)	AUC (%)
50	19.11	10.5	65.3
80	22.58	10.5	68.21
100	20.3	10.5	67.06
150	**27.59**	10.5	67.02
200	22.77	10.5	**68.34**
250	22.45	10.5	66.85
300	21.83	10.5	66.55
500	26.87	10.5	68.16
700	21.67	10.5	66.31

Tabla 9: Valores de AUC y AP para diferentes números de clusters

En este caso, aparte de utilizar el mismo número de clusters que en el apartado anterior, añadimos otros que son menores ya que el número de puntos característicos obtenidos por SIFT [31] es menor debido a que las imágenes son más pequeñas, obteniendo los resultados óptimos con k=150 y k=200 *clusters*. Observando los resultados y comparando con los que se obtienen en la Tabla 8, hemos empeorado un poco, esto puede ser debido a que como la cantidad de puntos ha disminuido, el diccionario visual también lo hace y otra posibilidad puede ser debido a que el fondo contiene información relevante.

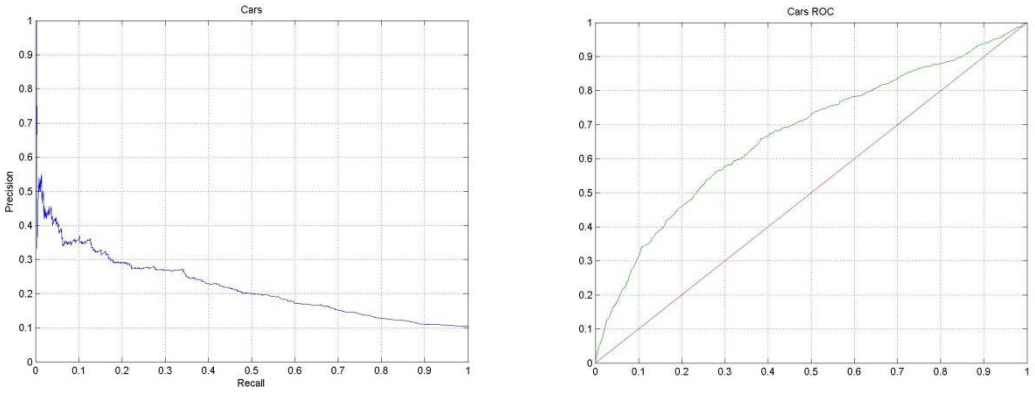


Ilustración 28: Gráficas Precision Recall (izqda.) y ROC (dcha.) para k=150

Visto que no alcanzamos unos resultados equiparables a los de la literatura, esto puede ser debido a varios factores:

1.- Uno de los factores más importantes y por lo que no podemos compararlos exactamente es debido a que los resultados están expuestos para una serie de imágenes de test diferente a la

nuestra, ya que nosotros utilizamos como imágenes de test las imágenes que están dadas como imágenes de validación.

2.- Segundo punto, es que la forma de caracterizar a las imágenes no es la misma, en nuestro diseño caracterizamos las imágenes aplicando el extractor SIFT, en algunos artículos como ya hemos visto anteriormente, SIFT sólo lo utilizan para obtener el descriptor del punto.

3.- Otro factor puede ser debido a que el método para crear el diccionario visual no es mediante *k-means* sino mediante GMM.

4.- Por último es posible que para entrenar los clasificadores utilicen más imágenes que nosotros, es decir, por ejemplo utilicen todas las imágenes que tienen disponibles para entrenar (imágenes de entrenamiento y validación) ya que las imágenes de test son diferentes.

Debido a estos factores, decidimos en primer lugar revisar el punto 2 descrito estudiando un método diferente de escoger los puntos característicos. Seguiremos una serie de pautas estudiadas en el artículo [7].

4.3 Mallado de las imágenes + SIFT

Este método ya explicado en el capítulo 3.1.2, se basa en escalar todas las imágenes, para que todas ellas tengan el mismo número de pixeles, pero sin cambiar su forma, es decir solo cambiamos el área de la imagen. Posteriormente se realiza un mallado de las imágenes y se concretan unas regiones a las que se les va a aplicar el descriptor SIFT [30], el procedimiento seguido es el que se muestra en la Ilustración 17.

Se realizan diferentes tipos de cuadrículas que proporcionan entre 500 y 5000 puntos, aplicando el extractor SIFT (ver capitulo 4.2) se obtienen de media 800 puntos característicos por imagen, por lo que estamos aumentando considerablemente el número de puntos característicos a la vez que conseguimos que todas las imágenes tengan el mismo número de puntos. Para cada uno de estos casos realizamos diferentes *codebook* con un número de palabras entre 500 y 2000.

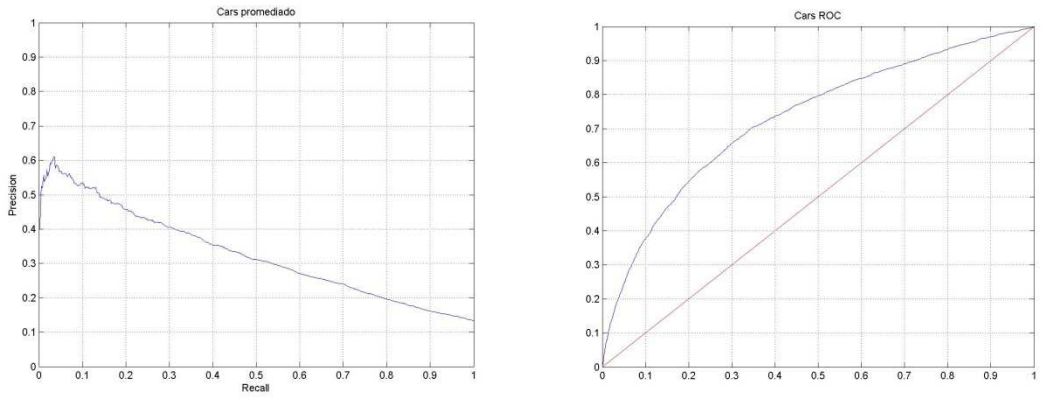


Ilustración 29: Gráficas Precision Recall (izqda.) y ROC (dcha.) para k=250 y número de puntos característicos=2000.

Las siguientes graficas de Precision y Recall y ROC, son curvas promediadas. Se realizan 10 pruebas diferentes para cada valor de k y para cada número de puntos característicos extraídos. En cada una de ellas el conjunto de entrenamiento se ha dividido de 10 formas diferentes (60 % para crear el *codebook* y un 40 % para entrenar el SVM). Las curvas que mostramos son un promedio sobre las 10 pruebas que hemos realizado.

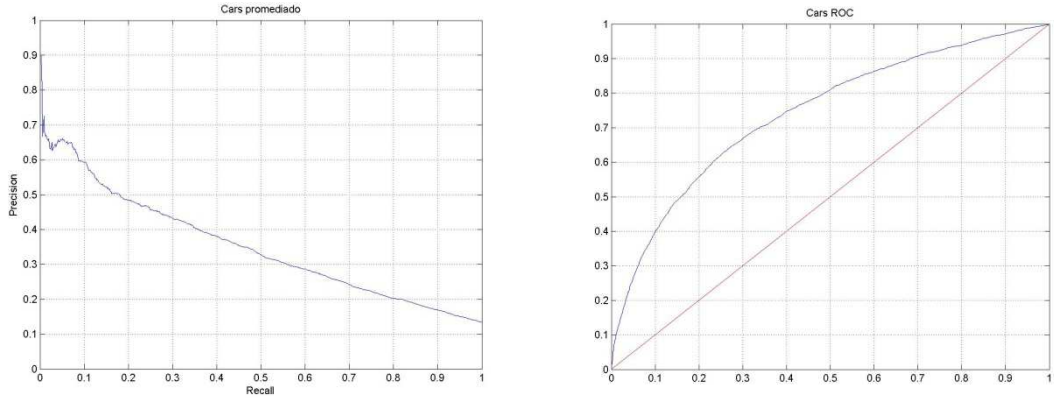


Ilustración 30: Gráficas Precision Recall (izqda.) y ROC (dcha.) para k=500 y número de puntos característicos=2000

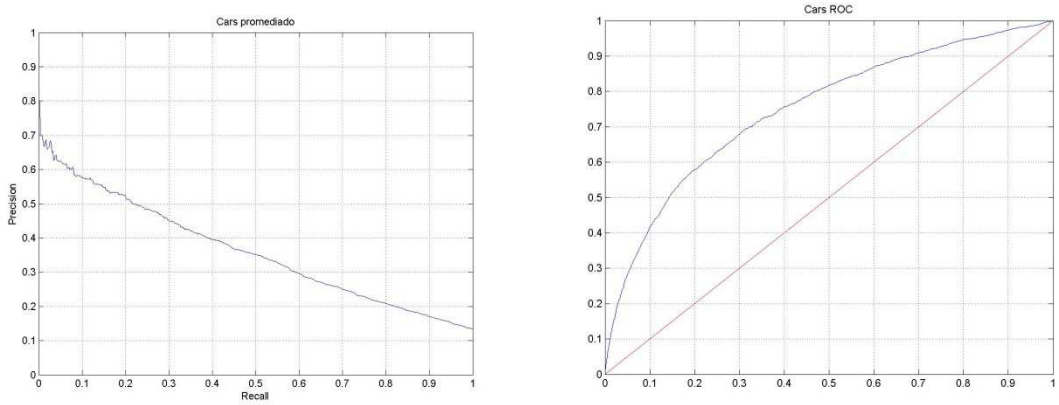


Ilustración 31: Graficas Precision Recall (izqda.) y ROC (dcha.) para k=1000 y número de puntos característicos=2000

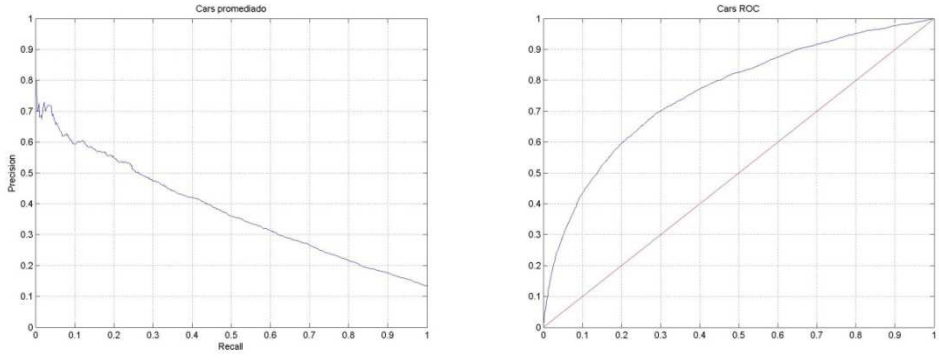


Ilustración 32: Gráficas Precision Recall (izqda.) y ROC (dcha.) para k=2000 y número de puntos característicos=2000

BOW Núm. puntos	k	AP (%)	AUC (%)
500	250	27.99	69.33
	500	29.48	69.82
	1000	31.41	70.92
	2000	31.17	71.14
1000	250	35.34	72.45
	500	34.59	73.68
	1000	36.78	74.80
	2000	37.20	74.88
2000	250	33.42	73.07
	500	37.77	74.24
	1000	37.8	75.02
	2000	**39.26**	**76.16**
5000	250	25.67	66.71
	500	29.01	72.19
	1000	31.77	73.81
	2000	33.74	73.32

Tabla 10: Valores de AUC y AP para diferentes números de clusters y diferente número de puntos característicos, resaltados los valores con mayor acierto. Imágenes ordenadas en el CD adjunto.

Observando los resultados que encontramos en la Tabla 10, comprobamos que hemos mejorado los resultados, a medida que aumentamos el número de puntos característicos mejoramos los resultados aunque también se llega a un límite donde se vuelven a empeorar como podemos comprobar con 5000 puntos característicos.

Como vemos con 2000 puntos característicos extraídos y k=2000 *clusters* es donde mejores resultados obtenemos: Una Precisión media del 39.26 % y una área bajo la curva ROC del 76.16%. Estos resultados mejoran en comparación a los resultados obtenidos en el capitulo 4.2.1 en más de 7 puntos de AP y en casi 5 puntos el AUC. También comprobamos que a medida que aumentamos el número de *clusters*, sin importar el número de puntos extraídos, los resultados mejoran. Pese a la mejora de los resultados, seguimos obteniendo tasas por debajo a los de la literatura, por lo que vamos a probar otra estrategia como es la de reducir la dimensión utilizando PCA.

- **PCA con los puntos característicos obtenidos mediante el mallado de la imagen**

En este apartado aplicaremos reducción de dimensionalidad con el método PCA, reduciendo la dimensión a 100 descriptores, esta reducción de la dimensión nos proporciona varios beneficios como son la disminución del coste computacional para la creación del diccionario visual con k-means y la reducción del posible ruido o de valores que no signifiquen nada.

Esta reducción de dimensionalidad la aplicamos únicamente sobre los 1000 puntos extraídos sobre la cuadricula.

BOW + PCA (k)	AP (%)	AUC (%)
250	24.52	66.80
500	29.23	71.84
1000	**31.06**	**72.28**
2000	29.69	71.55

Tabla 11: Valores de AUC y AP para BOW + PCA para 1000 puntos característicos. Imágenes ordenadas en el CD adjunto.

La Tabla 11 hay que compararla con los valores obtenidos en el apartado anterior para un mallado de 1000 puntos, y como se observa tras hacer la reducción de dimensionalidad los resultados empeoran ligeramente. A pesar de que en la literatura se suele utilizar PCA para reducir la dimensión de los vectores antes de clasificarlos, observamos que en nuestras pruebas la pérdida de información debida a la reducción de la dimensión nos da como resultados peores tasas de precisión media y de área bajo la curva ROC.

4.4 Otros métodos de clasificación

En este apartado estudiaremos otros métodos de clasificación, ya no se utilizará el clasificador SVM. Algunos de los métodos que implementaremos ya no necesitarán de la creación de un diccionario visual como es el caso de NBNN [6].

4.4.1 NBNN

Mediante la implementación del método NBNN [6] obtuvimos diferentes resultados, según el número de imágenes de entrenamiento que utilizáramos para clasificar las imágenes de test.

Después de obtener todos los puntos característicos de las imágenes de entrenamiento, mediante la extracción de puntos característicos utilizando SIFT al igual que en el capitulo 4.2.1, los puntos extraídos los dividimos de tres maneras diferentes, esto es cogiendo más puntos característicos de clase positiva (existencia del objeto) que de clase negativa (no existencia del objeto), equiparando las clases y por ultimo cogiendo más puntos de clase negativa que de clase positiva, obteniendo así tres resultados diferentes.

En las tres formas explicadas, todas las bases de datos creadas con las imágenes de entrenamiento se igualaron clases, es decir, habrá el mismo número de imágenes de clase positiva (existencia del objeto) que de clase negativa (no existencia del objeto).

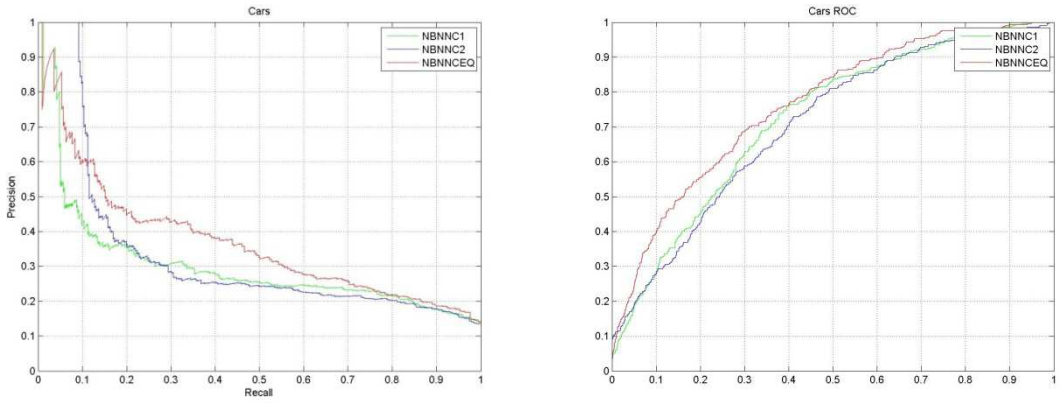


Ilustración 33: Gráfica de Precision vs Recall (izqda.) Curvas ROC (dcha.)

En la Ilustración 33 observamos las gráficas de Precision frente a Recall para la detección de coches. Se muestran 3 curvas diferentes, cada una de ellas representa una manera de escoger los puntos en la parte de entrenamiento, es decir, cada una de ellas representa la forma en la que se ha dividido las imágenes de entrenamiento. La Tabla 12 muestra los valores de AP y AUC para cada una de las diferentes agrupaciones de puntos característicos y como podemos comprobar los mejores valores los obtenemos cuando las clases (positiva y negativa) están igualadas.

NBNN	AP (%)	AUC (%)
Clases igualadas	39.03	76.27
Clase 1 > Clase -1	33.28	72.38
Clase 1 < Clase -1	35.78	70.82

Tabla 12: Valores de AUC y AP para diferentes formas de agrupar los puntos de entrenamiento. Imágenes ordenadas en el CD adjunto.

Si comparamos estos resultados con los obtenidos en la Tabla 8 del apartado 4.2.1 vemos que estos son mejores. Además estos resultados son comparables a los obtenidos en la Tabla 10 para el mejor de los casos. Teniendo en cuenta que en dicha tabla se han ajustado parámetros como es el número de puntos extraídos. Podemos concluir que el método NBNN funciona mejor para este problema.

Aun así nuestros resultados siguen estando por debajo del estado del arte. Es posible que los resultados mejoren al usar mas imágenes de entrenamiento, como hacen en el desafío Pascal, ya que usan para entrenar tanto las imágenes de entrenamiento como las de validación (el test se realiza sobre otro conjunto). Vamos a estudiar esto haciendo un tipo de validación llamada leave-one-out, que consiste en escoger una imagen de validación como imagen de test y el resto como imágenes de entrenamiento.

Solo se realizan pruebas igualando el número de puntos característicos en las dos clases, ya que es la prueba con la que obtenemos mejores resultados en el experimento anterior. La Ilustración 34 nos muestra las graficas de P/R y la curva ROC para esta prueba, y en la Tabla 13 resumimos los valores de AP y AUC bajo la ROC de estas figuras.

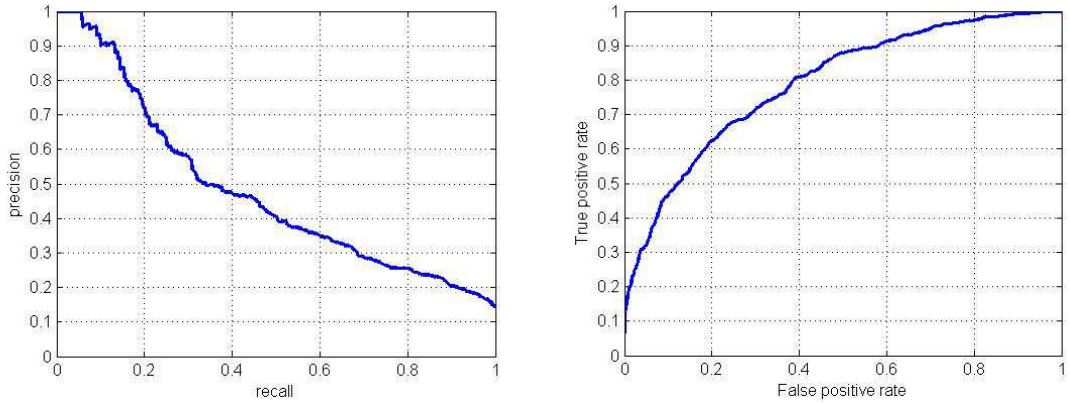


Ilustración 34: Precision/Recall y ROC para imágenes igualando puntos característicos en las clases

NBNN	AP (%)	Priori (%)	AUC (%)
Clases igualadas	48.46	14	79.01

Tabla 13: Valores de AUC y AP para diferentes formas de agrupar los puntos. Imágenes ordenadas en el CD adjunto.

Se observa una clara mejora de los resultados para las clases igualadas en cuanto a AP y AUC de la curva ROC.

Posteriormente, se decidió comprobar como funcionaria este método, si las imágenes las recortábamos quedándonos solo con los objetos de interés, es decir, eliminando el fondo, como hicimos con el método de BOW+SVM descrito en el capitulo 4.2.2. La Tabla 14 y la Ilustración 35 nos muestran los resultados para esta prueba.

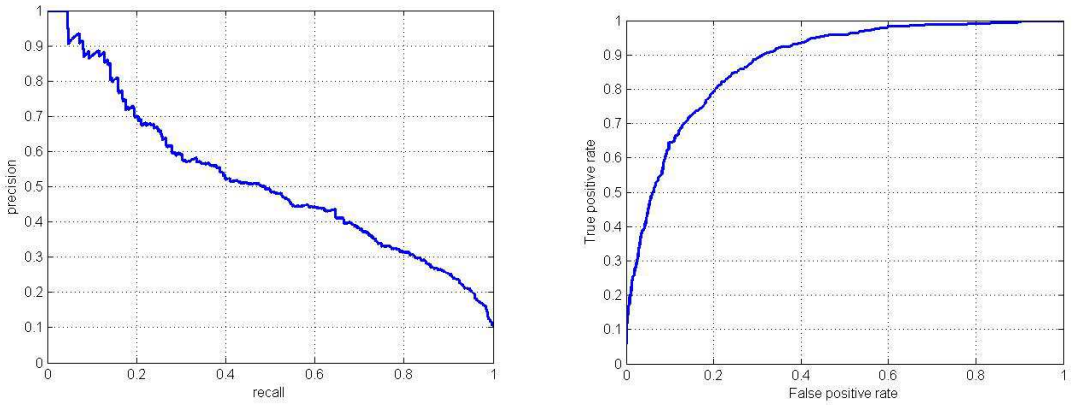


Ilustración 35: Gráfica de Precision vs Recall (izqda.) Curvas ROC (dcha.) para imágenes recortadas

NBNN Box	AP (%)	Priori (%)	AUC (%)
Clases igualadas	51.73	10.5	87.81

Tabla 14: Valores de AUC y AP para diferentes formas de agrupar los puntos de entrenamiento para imágenes recortadas. Imágenes ordenadas en el CD adjunto.

Observando la Ilustración 35 y la Tabla 14 comprobamos que los resultados mejoran sustancialmente al utilizar las imágenes recortadas, es decir al igual que en el capitulo 4.2.2 descartando el fondo. Hay que tener en cuenta además que el problema es más difícil ya que tras recortar el priori baja de un 14% a un 10%.

Si comparamos los resultados obtenidos con los de la Tabla 9 podemos concluir que el empeoramiento de los resultados al recortar las imágenes mediante BOW+SVM debe ser debido a que el menor número de puntos extraídos dificulta la creación del diccionario y no a que el fondo contenga información relevante.

A la vista de los resultados obtenidos hasta ahora podemos deducir varias conclusiones:

- En igualdad de condiciones, extrayendo los puntos característicos con SIFT, NBNN obtiene mejores resultados que BOW+SVM.
- Los mejores resultados de BOW+SVM (ajustando el número de puntos mediante el mallado y el número de palabras en el diccionario) son equiparables a NBNN. Esperamos que si aumentamos el número de puntos utilizando el mallado NBNN mejorara sus resultados aunque estas pruebas no se han realizado.
- Utilizando el método *Leave-One-Out* y así aumentando el número de imágenes de entrenamiento se mejoran sustancialmente los resultados de NBNN (3% en AUC y 9% en AP). Este mismo método se ha utilizado con BOW+SVM pero los resultados no mejoran.
- Al recortar las imágenes para quedarnos sólo con las regiones de interés que contienen a los objetos, mejoramos aún más los resultados con un 87% de AUC bajo la curva ROC. Esto con BOW+SVM no ocurre, debido probablemente a que el diccionario no es tan bueno al haber disminuido el número de puntos característicos.
- Los mejores resultados que se describen en la literatura son con el modelo BOW+SVM, y mejoran notablemente los aquí expuestos. Dado que nuestros resultados no mejoran los resultados de la literatura debemos concluir que puede ser debido a factores como un preprocesado de las imágenes diferente al nuestro o a la forma de construir los diccionarios. Esta aparente falta de rendimiento de BOW en comparación con otros métodos ha sido también descrita por otros autores [38; 28].

4.5 NBNN con base de datos Visual Geometry Group

En este apartado, y después de haber visto todos los resultados que hemos ido obteniendo con los diferentes métodos de clasificación, podremos decir, en resumen que aunque en la literatura se presenta el método BOW+SVM como de los mejores detectores de objetos en imágenes mediante la extracción de puntos característicos [7; 5], se observa que, aunque sea una algoritmo más simple y con menos coste computacional, los mejores resultados los obtenemos con el algoritmo NBNN [6; 38]. Por eso se va a comprobar si con la base de datos de Visual Geometry Group también mejoran los resultados comparándolos con los que obtuvimos con el BOW+SVM.

La Ilustración 36 y la Tabla 15 nos muestran los resultados para el método NBNN. En este caso sólo se realizará la prueba con los puntos igualados, ya que han sido la que mejores resultados nos han proporcionado en anteriores pruebas.

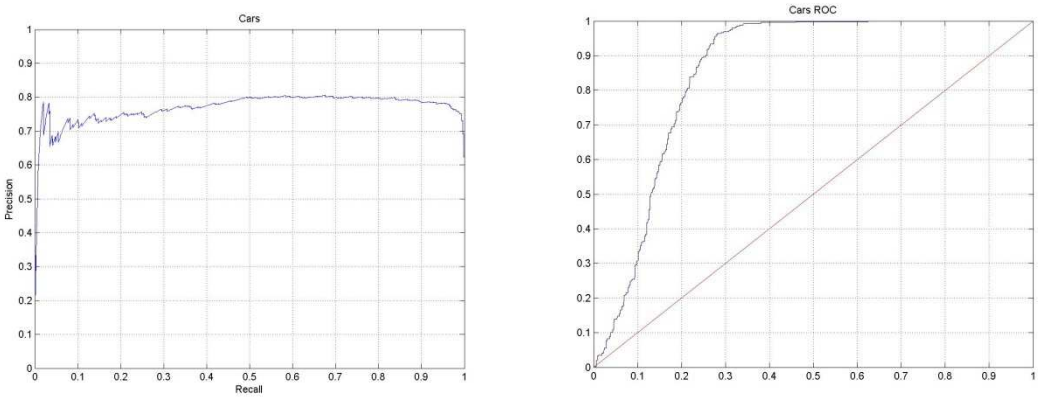


Ilustración 36: Gráfica Precision vs Recall (izqda.) y ROC (dcha.)

NBNN	AP (%)	AUC (%)
Clases igualadas	78.62	85.68

Tabla 15: Valores de AP y AUC para NBNN con la base de datos de Visual Geometry Group. Imágenes ordenadas en el CD adjunto.

Comparando estos resultados con la Tabla 6, vemos que para esta base de datos también mejoran ligeramente los valores de precisión media y del área bajo la curva ROC.

4.6 Pruebas para diferenciar entre objetos

En este apartado vamos a ver un estudio en el que se va a experimentar la detección de un objeto con respecto a otro, es decir, vamos a diferenciar entre coche y autobús, o coche y persona, etc.

Vamos a ver algunos ejemplos y sus correspondientes curvas de Precisión y recall. En todas los ejemplos que vamos a ver el método que hemos seguido ha sido la de extraer los puntos característicos mediante el algoritmo de SIFT [31] de los objetos de las imágenes, es decir utilizaremos las imágenes recortadas como en el capitulo 4.2.2. Posteriormente se creará un diccionario visual mediante la separación de los archivos de entrenamiento en un 60% y un 40%, y finalmente la aplicación del clasificador SVM. En las siguientes graficas el elemento nombrado primero es el de clase positiva y el segundo será el de clase negativa.

i. Coche – Autobús

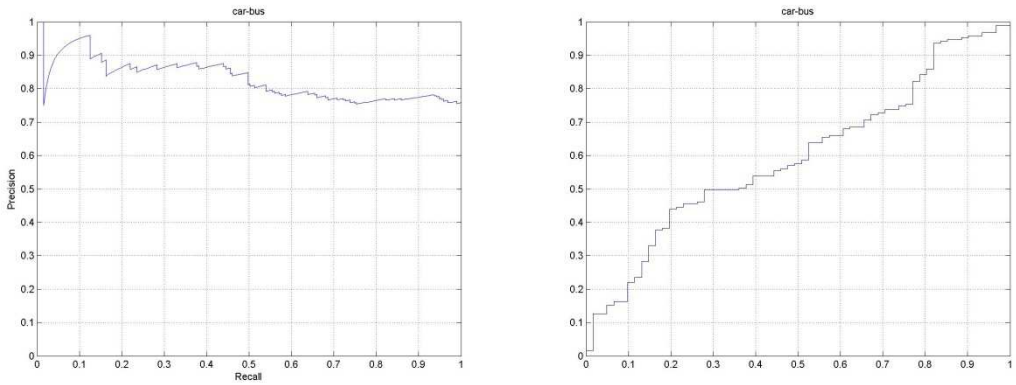


Ilustración 37: Curva Precision-recall y ROC para la detección de coches frente a los autobuses con BOW

ii. Coche – Avión

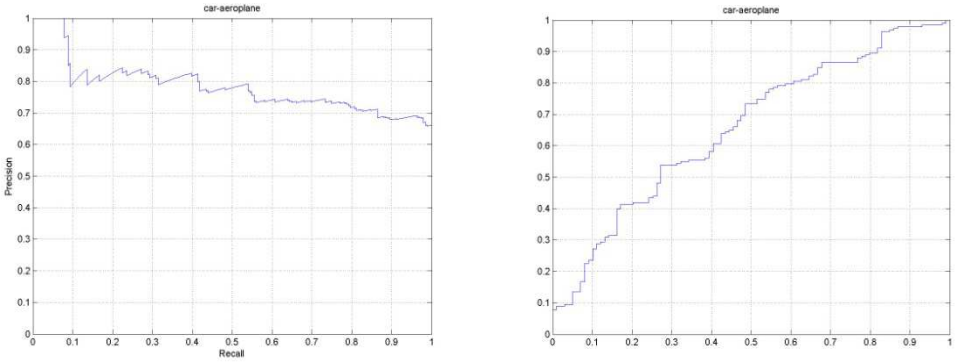


Ilustración 38: Curva Precision-recall y ROC para la detección de coches frente a los aviones con BOW

iii. Coche – Perro

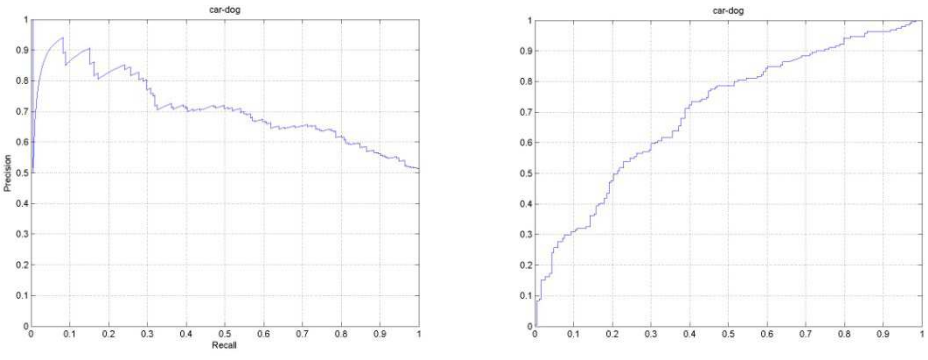


Ilustración 39: Curva Precision-recall y ROC para la detección de coches frente a perros con BOW

iv. Coche – Persona

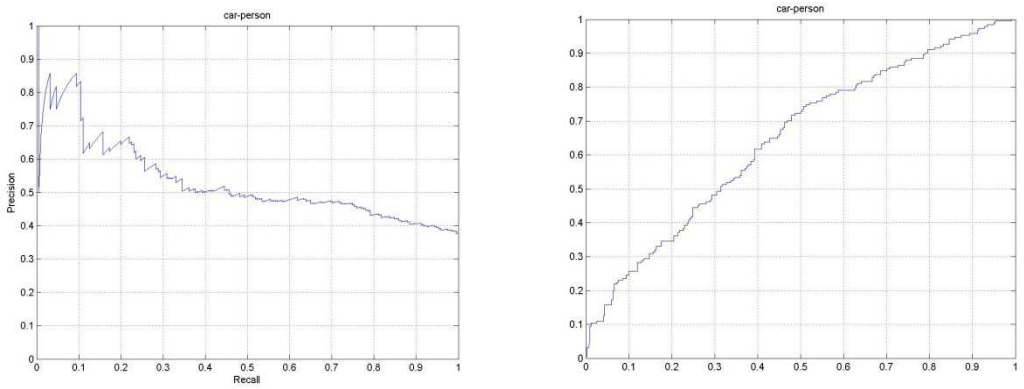


Ilustración 40: Curva Precision-recall y ROC para la detección de coches frente a personas con BOW

v. Gatos-Perros

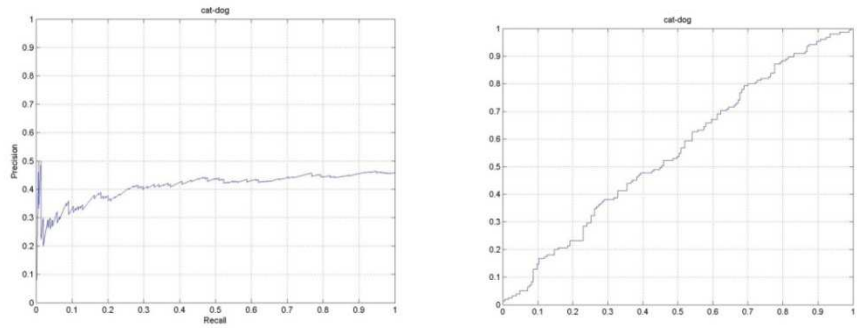


Ilustración 41: Curva Precision-recall y ROC para la detección de gatos frente a perros con BOW

En la siguiente tabla que mostramos, observamos los valores tanto de la precisión media como de del área bajo la curva (AUC).Como podemos comprobar en las gráficas los objetos que se diferencia mucho entre ellos se obtienen mejores resultados que en los que se aparecen como puede ser un perro y un gato.

BOW	AP (%)	Priori (%)	AUC (%)
Coche-Bus	84.54	75.79	59.54
Coche-Avión	78.99	65.86	65.53
Coche-Perro	72.65	51.07	70.31
Coche-Persona	56.79	37.52	64.68
Gato-Perro	46.66	45.86	55.37

Tabla 16: Valores de AP y AUC para diferentes comparaciones entre objetos y con el respectivo valor de la probabilidad a priori de coches o en el caso de gatos y de perros de gatos. Imágenes ordenadas en el CD adjunto.

Aplicaremos el método NBNN para realizar el mismo estudio que hemos realizado con BOW. Los resultados se muestran en las siguientes figuras. En la Tabla 17 se muestran los diferentes valores de AP y AUC para los correspondientes emparejamientos de clasificación con sus correspondientes probabilidades a priori.

i. Coche – Autobús

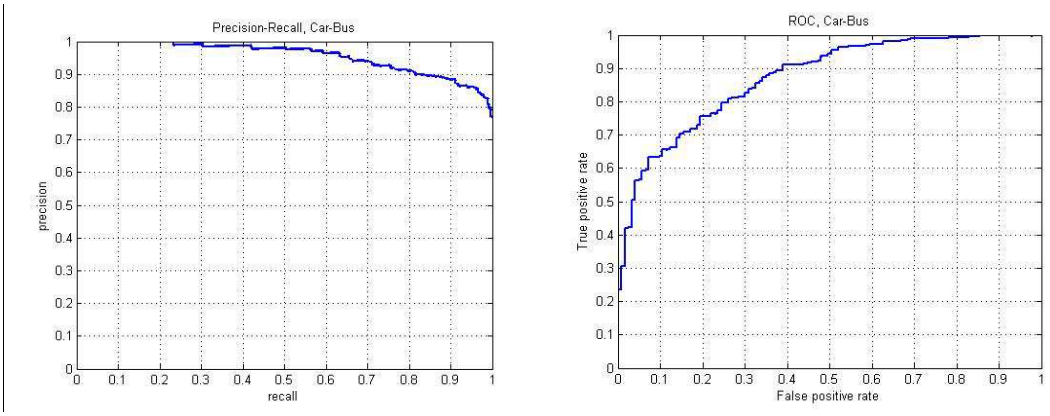


Ilustración 42: Curva Precision-recall y ROC para la detección de coches frente a los autobuses con NBNN

ii. Coche – Avión

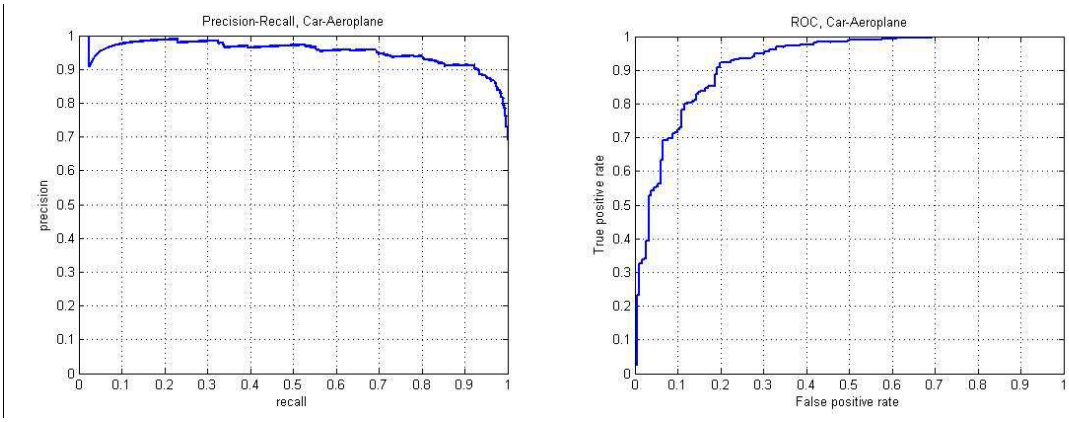


Ilustración 43: Curva Precision-recall y ROC para la detección de coches frente a los aviones con NBNN

iii. Coche – Perro

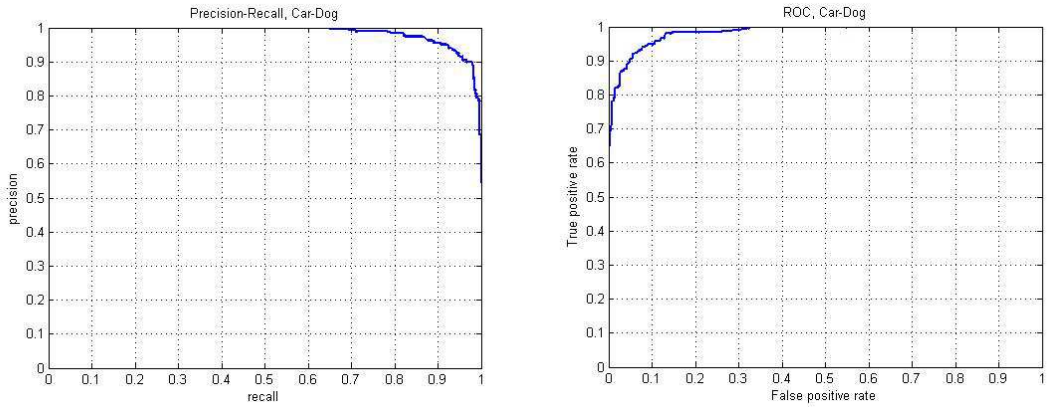


Ilustración 44: Precision-recall y ROC para la detección de coches frente a perros con NBNN

iv. Coche – Persona

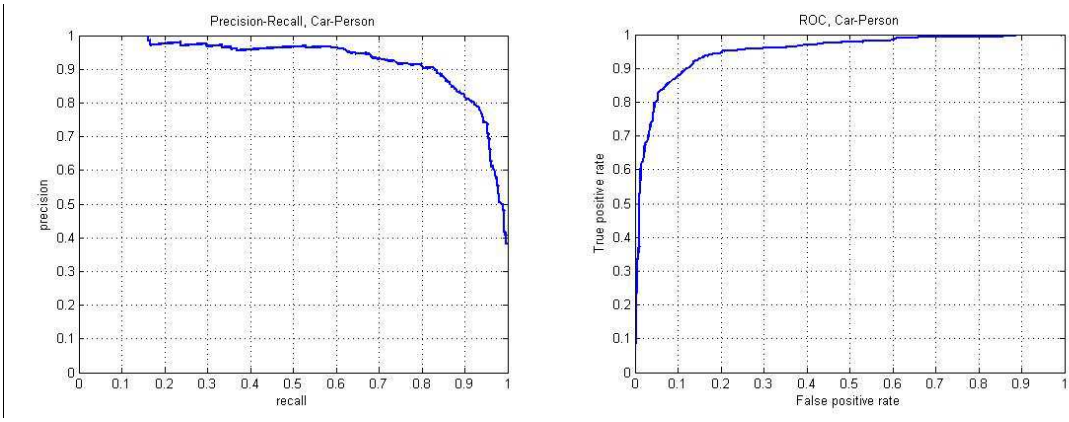


Ilustración 45: Curva Precision-recall y ROC para la detección de coches frente a personas con NBNN

v. Gatos - Perros

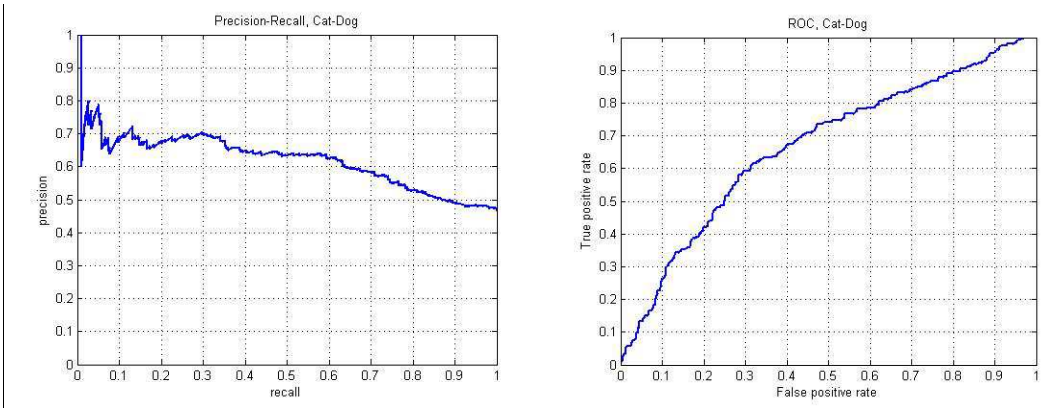


Ilustración 46: Curva Precision-recall y ROC para la detección de gatos frente a perros con NBNN

NBNN	AP (%)	Priori (%)	AUC (%)
Coche-Bus	94.88	76.75	87.08
Coche-Avión	94.59	69.05	92.07
Coche-Perro	96.55	54.57	98.29
Coche-Persona	89.95	38.09	94.93
Gato-Perro	64.8	46.6	66.62

Tabla 17: Valores de AP y AUC para diferentes comparaciones entre objetos y con el respectivo valor de la probabilidad a priori de coches o en el caso de perros y de gatos de gatos para imágenes recortadas. Imágenes ordenadas en el CD adjunto.

De nuevo se comprueba que para todos los casos NBNN funciona mejor que BOW+SVM si comparamos la Tabla 17 con la Tabla 16. Nótese que los prioris en ambas tablas son ligeramente diferentes debido a que las divisiones en entrenamiento y test se realizan de forma diferente.

Observando los resultados obtenidos mediante los distintos métodos que hemos utilizado, nos decantaremos por el algoritmo clasificador NBNN como clasificador de nuestro sistema final, ya que son los mejores resultados que obtenemos.

Adicionalmente añadimos una prueba con la que comparar estos últimos resultados de NBNN. Pero esta vez con la imagen al completo sin recortar, pero solo utilizamos como imágenes positivas aquellas que tienen solo coche y como imágenes negativas aquellas que contienen el otro objeto pero no contienen coche. En la Tabla 18 mostramos los resultados con sus respectivos valores de las probabilidades a priori. Comparando ambas tablas, observamos que mejoramos en todos los casos, por ello decidiremos en nuestro sistema final utilizar el método de recorte de las imágenes para clasificar.

NBNN	AP (%)	Priori (%)	AUC (%)
Coche-Bus	94.36	77	73.17
Coche-Avión	93.64	88	89.08
Coche-Perro	92.93	66	91.54
Coche-Persona	66.78	20	86.55
Gato-Perro	58.73	45	62.32

Tabla 18: Valores de AP y AUC para diferentes comparaciones entre objetos y con el respectivo valor de la probabilidad a priori de coches o en el caso de perros y de gatos de gatos para imágenes sin recortar.

4.7 Base de datos privada

En este apartado mostraremos los resultados obtenidos con la base de datos proporcionada por la empresa privada, aplicando el mejor método de clasificación que nos ha otorgado mejores resultados en las pruebas anteriores, NBNN.

Para nuestro diseño final recortaremos las imágenes, ya que NBNN funciona mejor con las imágenes recortadas. El fondo en todas las imágenes es el mismo por lo que no nos aportara información relevante y por último partimos de la suposición de que tenemos un segmentador capaz de separar el objeto del fondo.

Dado que tenemos muy pocas imágenes, utilizaremos el planteamiento *leave-one-out*. En total hay 22 imágenes, de las cuales 12 son coches y 10 son autobuses. El método que utilizamos para extraer los puntos característicos es SIFT sin aplicar mallado. Los resultados se muestran en la Ilustración 47 y en la Tabla 19, observamos que son unas curvas ROC y P/R perfectas, lo que nos indica que es posible detectar todos los casos positivos sin introducir ningún falso positivo. En la Ilustración 48 observamos como quedan las imágenes después de haber sido clasificadas y ordenadas según su cercanía a la clase positiva, es decir la primera imagen es la que más cercana está de ser un coche y la última imagen de ser la más alejada a ser un coche.

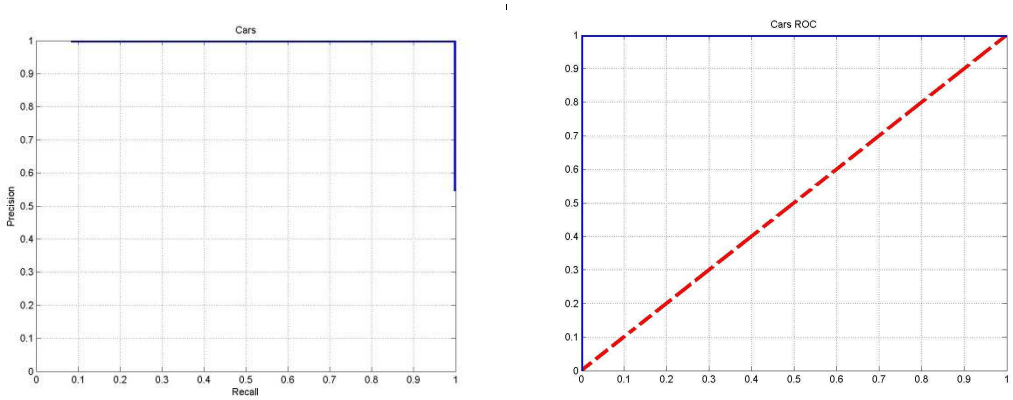


Ilustración 47: Gráficas Precision/Recall y ROC para imágenes recortadas

NBNN	AP (%)	AUC (%)
Coche	100	100

Tabla 19: Valores de AP y AUC para NBNN con las imágenes recortadas

Ilustración 48: Imágenes ordenadas después de haber sido clasificadas mediante el método NBNN

En conclusión, el sistema descrito es capaz de diferenciar entre coches y autobuses de una manera perfecta dentro de un entorno controlado, suponiendo que tenemos un segmentador ideal que nos separe perfectamente el objeto del fondo. Hay que tener en cuenta que se tiene poca variedad en los datos, por lo que se espera que cualquier alteración en alguno de estos factores pueda empeorar la calidad de los resultados.

5. Conclusiones y trabajo futuro

5.1 Conclusiones

Durante la elaboración de este proyecto se han llevado a cabo diferentes pruebas para la detección de objetos en imágenes mediante la extracción de puntos característicos. Cada sistema utiliza o diferente información o diferente etapas en su elaboración.

5.1.1 BOW+SVM

En esta etapa comentaremos los resultados obtenidos con el método de BOW+SVM. En primer lugar, se aplico el método de BOW+SVM tanto a la base de datos de Visual Geometry Group como a la proporcionada por Pascal 2007. En esta primera prueba se utilizaron como puntos característicos los extraídos por el algoritmo SIFT [31] obteniendo una media de 800 puntos característicos por imagen. Se aplicó un agrupamiento de características utilizando *k-means* con k=1000 *clusters* y utilizando SVM como clasificador.

Para la base de datos de Visual Geometry Group, obtenemos como resultados un área bajo la curva ROC del 83.89% y una Precisión media del 78.5%. Siendo unos resultados satisfactorios, no alcanzamos los resultados que se muestran en la literatura. En nuestro caso para un recall del 97.7% tenemos una precisión del 70%, cuando en el artículo [13] se muestra una precisión del 98%. Esto puede ser debido a la forma de dividir las imágenes en entrenamiento y en test, en el artículo no se menciona como se realiza esta división, por lo que nosotros hemos realizado una división aleatoria de las imágenes como explicamos en el capitulo 3.6.2. También puede ser debido a diferencias en el preprocesado de las imágenes, en la extracción de los puntos característicos o en la forma de construir el diccionario.

Para la base de datos de Pascal 2007 se obtuvo un área bajo la curva ROC del 71% y una precisión media del 28.17%. Para esta base de datos, los resultados que se exponen son para un grupo de imágenes de test diferente al nuestro, en este caso para un recall del 97.7% obtenemos una precisión del 14%, cuando en el desafío los 5 mejores algoritmos tienen una precisión del 20%. Esto puede ser debido a varios factores, como ya hemos dicho el grupo imágenes de test es diferente al que utilizamos, por lo que seguramente para entrenar ellos utilicen tanto las imágenes de entrenamiento como de validación, por lo que tienen una mayor número de ejemplos para diferenciar entre objetos. El preprocesado de las imágenes puede ser diferente, al igual que la extracción de los puntos característicos. Y finalmente la forma de crear el diccionario.

Para intentar alcanzar los resultados en Pascal 2007 realizamos diferentes pruebas, variando algunos factores como es el número de *clusters*, el preprocesado de las imágenes, la extracción de puntos característicos.

Se varia el número de *clusters*, con lo que comprobamos que a medida que aumentamos el número de centroides mejora la precisión media, hasta un punto en que el que vuelve a caer el valor, esto nos dice que hay una relación entre el número de puntos extraídos y el número de *clusters* en que se divide el conjunto de datos.

Aprovechando que se adjuntan a esta base de datos (Pascal 2007) una serie de anotaciones de las imágenes, se recortan y así eliminamos el fondo de la imagen. Se realizan pruebas variando el número de *clusters* ya que en este caso el número de puntos es menor. Al no mejorar los resultados que obtuvimos sin recortar podemos concluir que posiblemente el fondo proporcione información como para clasificar las imágenes, o que al disminuir el número de puntos los diccionarios empeoren y no se realice un agrupamiento bueno.

Viendo que variando el número de *clusters* o recortando las imágenes no conseguimos mejorar, es posible que al realizar un preprocesado de las imágenes se consiga mejorar. En este caso

realizaremos un mallado de las imágenes y aplicaremos SIFT sobre este mallado para obtener el descriptor de los puntos. Se realizaran diferentes mallados obteniendo entre 500 y 5000 puntos característicos por imagen. Al aumentar el número de puntos también se aumentara el numero de *clusters* variando entre 500 y 2000 *clusters*. Observamos que al aumentar el número de puntos los resultados mejoran, siendo el mejor caso con 2000 puntos característicos y 2000 *clusters* con una precisión media del 39.26% y un área bajo la curva ROC del 76.16%. Esto mejora en casi 5 puntos (AUC) y en más de 7 puntos (AP) a los resultados de la sección 4.2.1.

Se observa también en estas pruebas que la elección de los puntos característicos es importante a la hora de clasificar, ya que algunos puntos contienen mas información que otros, por eso se decide seleccionar los puntos mediante un mallado de la imagen, teniendo así el mismo número de puntos característicos en todas las imágenes.

Viendo que con este método de clasificación de imágenes no alcanzamos los resultados de Pascal 2007, se prueba con otro clasificador.

5.1.2 **NBNN**

En este apartado analizaremos los resultados obtenidos con el método NBNN.

Siguiendo el mismo proceso que en BOW+SVM, primero extraeremos los puntos característicos utilizando SIFT. Obteniendo una precisión media del 39.03 y un área bajo la curva del 76.27% en el caso en el que las clases (positiva y negativa) de las imágenes de entrenamiento están igualadas. Estos resultados superan a BOW+SVM en casi 8 puntos (AP) y en 5 puntos (AUC) para el mejor resultado obtenido con este proceso. Incluso NBNN iguala el mejor resultado que se obtiene con el método BOW+SVM, habiendo realizado en BOW un ajuste de los parámetros como es el preprocesado de las imágenes y variando el número de clusters. Podemos concluir que NBNN mejora sustancialmente a BOW+SVM en este problema.

Aun así los resultados no alcanzan los valores del estado del arte, por lo que se decide realizar una prueba llamada Leave-One-Out. En este caso utilizaremos más imágenes de entrenamiento y una sola imagen de validación como test. Se observa una clara mejora de los resultados, mejorando tanto AP en casi 9 puntos y AUC en 3 puntos.

También se realizaron pruebas con las imágenes recortadas, al igual que con BOW+SVM. Comprobamos que mejoramos nuestros resultados, y también volvemos a comprobar que mejora a los reportados por BOW, por lo que se puede descartar la posibilidad que mencionamos anteriormente en el que el fondo contenga información que nos haga empeorar los resultados.

Visto que los mejores resultados que obtenemos es con NBNN, decidimos comprobar si esto también funciona con la base de datos de Visual Geometry Group. En este caso también mejoran los resultados.

5.1.3 **Conclusiones finales**

Podemos concluir según los resultados vistos, que en igualdad de condiciones, misma forma de extraer los puntos característicos, NBNN funciona mucho mejor que BOW+SVM para nuestro problema. Incluso los mejores resultados de BOW+SVM, ajustando el número de puntos característicos (mallado) o ajustando el numero de palabras en el diccionario, son equiparables a NBNN sin ajustar ningún parámetro.

Si aumentamos el número de imágenes para entrenar, utilizando el método LOO (Leave-One-Out), los resultados mejoran sustancialmente. Por lo que podemos suponer que en el desafío Pascal 2007, a la hora de entrenar utilizan tanto las imágenes de entrenamiento como las imágenes de validación, y de test otro grupo de imágenes. Para BOW+SVM también se aplico este método pero los resultados no mejoraron lo que ya teníamos con BOW, realizando el mallado.

Por último, recortando las imágenes para quedarnos solo con las regiones de interés de los objetos y descartando el fondo, otorga mejores resultados NBNN, obteniendo un AUC del 87% y una AP del 51.73%, esto no ocurre con BOW+SVM. Esto puede ser debido a un deterioro en la creación del diccionario debido a que hay un menor número de puntos.

Aunque los mejores resultados reportados en la literatura sean con BOW+SVM y que mejoran notablemente a los que exponemos aquí, debemos concluir que la calidad de los resultados de la literatura se deban a factores que no se explican, como el preprocesado de las imágenes, la extracción de puntos característicos, la forma de construir los diccionarios, o la utilización de un mayor número de imágenes de entrenamiento.

Ya que el mejor resultado que se obtiene es con NBNN y con las imágenes recortadas, se decide utilizar este método en un entorno más controlado y en que sólo tengamos que distinguir entre dos objetos, coche o autobús. Se trata de una base de datos privada proporcionada por una empresa de seguridad. Se obtienen resultados muy satisfactorios dentro de este entorno con una AP del 100% y una AUC del 100%. Como podemos observar las imágenes han sido totalmente clasificadas, es decir el sistema que hemos elaborado es capaz de diferenciar entre coche autobuses. Dentro de un entorno muy especifico en el que el fondo es el mismo, suponiendo que tenemos un segmentador muy bueno que separa el objeto del fondo a la perfección, y que tenemos poca variedad en los datos. Todas estas condiciones son optimas para que nuestro resultado sea tan bueno, esperamos que si varían estos parámetros los resultados empeoren.

5.2 Trabajo futuro

Una posible prueba futura seria la utilización de NBNN con los puntos característicos obtenidos tras aplicar el mallado de la imagen, como hemos visto este método mejoro los resultados de BOW, por lo que se espera que con NBNN también mejoren.

Por otro lado, se podría probar un diferente preprocesado de las imágenes, en nuestro caso realizamos un mallado regular sobre todas las imágenes, se podría probar a realizar un mallado diferente, como por ejemplo ir escogiendo regiones de la imagen, y cada región dividirla en subregiones así hasta obtener el número de puntos deseados, o realizar un mallado que no sea cuadricular, etc.

Como se comprobó, habrá puntos característicos que contengan información más importante que otros puntos a la hora de clasificar las imágenes, por lo que también se propone intentar algún método para conseguir quedarnos con esos puntos característicos y desechar el resto.

Otra prueba que se puede realizar es la de incluir mas imágenes para entrenar, en nuestro caso, en el modelo de BOW utilizamos solo un 40% de las imágenes para entrenar el clasificador, suponemos que si aumentamos el número de imágenes mejoraremos el resultado. Para ello también tendremos que hacer más eficientes los algoritmos, como por ejemplo que la agrupación que realiza *k-means* no sea la óptima, sino una aproximación, pudiendo así reducir el tiempo.

Bibliografía

1. **Zisserman, Josef Sivic and Andrew.** *Video Google: A Text Retrieval Approach to Object Matching in Videos.* UK : Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set, 2003.

2. **With, Rob G.J. Wijnhoven and Peter H.N. de.** *Comparing Feature Matching for Object Categorization in Video Surveillance.* s.l. : Advanced Concepts for Intelligent Vision Systems, 2009.

3. **P. C. Cattin, H. Bay, L. Van Gool, and G. Székely.** *Retina mosaicing using local features.* s.l. : ETH Zurich - Computer Vision Laboratory, 2006.

4. **Se, Lowe and Little.** *Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks.* s.l. : The International Journal of Robotics Research, 2002.

5. **Everingham, M. and Van~Gool, L. and Williams, C. K. I. and Winn, J. and Zisserman, A.** The {PASCAL} {V}isual {O}bject {C}lasses {C}hallenge 2007 {(VOC2007)} {R}esults. [Online] 2007. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

6. **Boiman, O., Shechtman, E., Irani, M.** *In defense of Nearest-Neighbor based image classification.* s.l. : In IEEE Conference on Computer Vision and Pattern Recognition, 2008.

7. **Perronnin, Florent.** *Universal and Adapted Vocabularies for generic visual categorization.* s.l. : IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 2008.

8. **C.-W. Chang, M.-Y. Lin, H.-J. Harn, Y.-C. Harn, C.-H. Chen, K.-H. Tsai, and C.-H. Hwang.** *Automatic segmentation of abnormal cell nuclei from microscopic image analysis for cervical.* s.l. : Nano/Molecular Medicine and Engineering (NANOMED),, 2009.

9. **Samboal, María Martínez.** *Detección de células Cancerosas en imágenes.* s.l. : PFC UAM EPS, 2012.

10. **Se, S., Lowe, David G. and Little, J.** *Vision-based mobile robot localization and mapping using scale-invariant features.* s.l. : Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2001.

11. **S. Dummais, J. Platt, and D. Heckerman.** *Inductive learning algorithms and representations for text categorization.* s.l. : In Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management, 1998.

12. **Group, Visual Geometry.** [Online] http://www.robots.ox.ac.uk/~vgg/data.

13. **G. Csurka, C. R. Dance, L. Fan, J. Willamowski, C. Bray.** *Visual categorization with Bags of Keypoints.* Meylan, France : ECCV International Workshop on Statistical Learning in Computer Vision, 2004.

14. **A. Kumar, C. Sminchisescu.** *Support kernel Machines for object recognition.* s.l. : In ICCV, 2007.

15. **Lowe, David G.** *Object recognition from local sclae-invariant features.* s.l. : International Conference on Computer Vision, 1999.

16. **Wikipedia.** Curva Precision/Recall. [Online] http://en.wikipedia.org/wiki/Precision_and_recall.

17. —. Curvas ROC. [Online] http://es.wikipedia.org/wiki/Curva_ROC.

18. **Lowe, D.** *Distinctive image features from sclae-invariant keypoints.* s.l. : International Journal of Computer Vision, 2004.

19. **Stephens, Chris Harris & Mike.** *A combined corner and edge detector.* s.l. : In Fourth Alvey Vision Conference, 1988.

20. **Brady, S.M. Smith and J.M.** *SUSAN- a nem approach to low level image processing.* s.l. : International Journal of Computer Vision, (May 1997).

21. **H. Zhang, A. Berg, M. Maire and J. Malik.** *Svm-knn: discriminative nearest neighbor classification for visual category recognition.* s.l. : CVPR, 2006.

22. **T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman and A. Y. Wu.** *An efficient k-means clustering algorithm: Analysis and implementation.* s.l. : IEEE Trans. Pattern Analysisi and Machine Intelligence, 2002.

23. **Reynolds, Douglas.** *Gaussian Mixture Models.* USA : MIT Lincoln Labaratory.

24. **Triggs, F. Jurie and B.** *Creating efficient codebooks for visual recognition.* s.l. : In ICCV, 2005.

25. **A. Bosch, A. Zisserman and X. Munoz.** *Image classification using random forests and ferns.* s.l. : ICCV, 2007.

26. **X. Munoz and A. Bosch, A. Zisserman.** *Scene Classification via pLSA.* s.l. : In Proc. ECCV, 2006.

27. **F. Monay, P. Quelhas, D. Gatica-Perez and J.-M. Odobez.** *Constructing Visual Models with a Latent Space Approach.* s.l. : IDIAP , 2005.

28. **G. Martínez-Muñoz, W. Zhang, N. Payet, S. Todorovic, N. Larios, A. Yamamuro, D. Lytle, A. Moldenke, E. Mortensen, R. Paasch, L. Shapiro and T. G. Dietterich.** *Dictionary-Free Categorization of Very Similar Objects via Stacked Evidence Trees.* s.l. : In: CVPR, 2008.

29. **H. Bay, A. Ess, T. Tuytelaars and Luc Van Gool.** *Speeded-Up Robust Features (SURF).* Zurich : in Computer Vision - ECCV , 2006.

30. **Mikolajczyk.** Affine Detectors. [Online] http://www.robots.ox.ac.uk/~vgg/research/affine/.

31. **Lowe, David.** Codigo Matlab para SIFT. [Online] http://www.cs.ubc.ca/~lowe/keypoints/.

32. **Lowe, M.Brown and D.** *Invariant features from interest point groups.* BMVC : s.n., 2002.

33. **Schmid, K. Mikolajczyk and C.** *Scale and Affine Invariant Interest point Detectors.* s.l. : In IJCV, 2004.

34. **Jolliffe, I.T.** *Principal Component Analysis.* s.l. : Springer, 2002.

35. **T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman and A. Y. Wu.** *A local search approximation algorithm for k-means clustering.* s.l. : Computational Geometry: Theroy and Applications, 2004.

36. **Dempster, A., Laird, N., Rubin, D.** *Maximum Likelihood from Incomplete Data via the EMAlgorithm.* s.l. : Journal of the Royal Statistical.

37. **Wikipedia.** Machine_learning. [Online] http://en.wikipedia.org/wiki/.

38. **R.N. Rojas-Bello, L.F. Lago-Fernández, G. Martínez-Muñoz, M.A. Sánchez-Montañés.** *A Comparison of techniques for robust gender recognition.* Madrir : s.n., 2011.

39. **Burges, Christopher J.C.** *A tutorial on Support Vector Machines for Pattern Recognition.* s.l. : Data Mining and Knowledge Discovery, 1998.

40. **VapNik, V.** *The Nature of Statistical Learning Theory.* New York : Springer-Verlag, 1995.

41. **Vapnik, V.** *Statistical Learning Theory.* New York : John Wiley and Sons, Inc., 1998.

42. **Elkan, Charles.** *Nearest Neighbor Classification.* 2011.

43. **R.O.Duda, P.E.Heart and D.G.Stork.** *Parzen Window.*

44. **Egan, J.P.** *signal detection theory and ROC analysis.* New York : Academic Press, 1975.

45. **Caltech-101.** Base de datos Caltech-101. [Online] http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

46. **Caltech-256.** Base de datos Caltech-256. [Online] http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

47. **flickr.** flickr. [Online] http://www.flickr.com.

48. **M. Marszalek, C. Schmid, H. Harzallah and J. van de Weijer.** *Learning Representations for visual object class recognition.* France : s.n., 2007.

49. **V. Ferrari, L. Fevrier, F. Jurie, C. Schmid.** *Groups of Adjacent Contour Segments for Object Detection.* s.l. : INRIA, 2006.

50. **Dance, Florent Perronnin and Christopher.** *Fisher Kernels on visual vocabularies for image categorization.* s.l. : In CVPR, 2007.

51. **T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu.** kmeans. [Online] January 27, 2010. http://www.cs.umd.edu/~mount/Projects/KMeans/.

52. **Lin, Chih-Chung Chang and Chih-Jen.** LIBSVM -- A Library for Support Vector Machines. [Online] November 5, 2011. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

53. **M. Sánchez-Montañés, Luis Lago, Ana González.** *Métodos avanzados en aprendizaje artificial: Teoría y aplicaciones a problemas de predicción.* Madrid : Asignatura EPS UAM , 0809.

Glosario

BOW: Bag of Words

SIFT: Scale Invariant Feature Transform

LOO: Leave One Out

NBNN: Naïve Bayes Nearest-Neighbor

AP: Average Precission

AUC: Area Under Curve

SVM: Support Vector Machine

KNN: K-Nearest Neighbors

PCA: Principal Component Analysis

GMM: Gaussian Mixture Model

ROC: Receiver Operating Characteritics

Anexos

A Manual del programador

Código para ejecutar en Matlab.

NBNN y Bounding Box

```
%% Funcion principal para el claculo del algoritmo NBNN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion Distancias                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

addpath([cd '/Main/']);
distancia=([cd '/DistancesC1/']); %%C2 Eq
path=([cd '/TestRec/']);
archivo=load([path 'ClaseRealTest.mat']);

id_imag=archivo.obj(:,1);
obj=archivo.clase(:,2);



%% Clasifico imagenes de Train y Test
matlabpool close force;
matlabpool;
parfor i=1:size(id_imag,1)

 fprintf('iteracion : %d\n', i);
 clase1(id_imag(i,1)); %eqpnts clase2

end
matlabpool close force;


%% guardo las distancias promedio
Matriz_distPromedio=zeros(size(obj,1),5);
c=1;
for i=1:size(id_imag,1)
    indTest=find(archivo.obj(:,1)==id_imag(i,1));
    ind=find(archivo.clase(:,1)==id_imag(i,1));
    id_obj=archivo.clase(ind,2);
    for k=1:archivo.obj(indTest,2)

    test=load([distancia 'distTest' sprintf('%d', id_imag(i,1)) '_'
sprintf('%d', id_obj(k,1)) '.mat']);

    distmas=(1/size(test.dist_Promedio,1))*sum(test.dist_Promedio(:,3));
```

```
distmenos=(1/size(test.dist_Promedio,1))*sum(test.dist_Promedio(:,2));

    Matriz_distPromedio(c,1)= id_imag(i,1);
    Matriz_distPromedio(c,2)= id_obj(k,1);
    Matriz_distPromedio(c,3)= distmas;
    Matriz_distPromedio(c,4)= distmenos;
    Matriz_distPromedio(c,5)= (distmas-distmenos)/(distmas+distmenos);
    c=c+1;
    end
end

file='Matriz_distPromedioC1.mat'; %'Matriz_distPromedioC2
                                   %'Matriz_distPromedioC3
save(file,'Matriz_distPromedio');


%% Cogemos el doble de imagenes de train de clase 1, con mas puntos de
%% clase 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion clase1                                                    %%
%%% Argumento de entrada: Nombre de la imagen                         %%
%%% Salida: Distancias de test a train                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function clase1(imagenTest)

addpath([cd '/Main/']);
distancia=([cd '/DistancesC1/']);
TrainRec=([cd '/TrainRec/']);
Test=([cd '/TestRec/']);


objTest=load([Test 'ClaseRealTest.mat']);
indTest=find(objTest.obj(:,1)==imagenTest);
ind=find(objTest.clase(:,1)==imagenTest);
id_obj=objTest.clase(ind,2);

obj=load([TrainRec 'ClaseRealTrain.mat']);


indclass1=find(obj.clase(:,3)==1);
indclass2=find(obj.clase(:,3)==-1);

if(size(indclass1,1)>size(indclass2,1))
    indclass1=indclass1(1:size(indclass2,1),:);
    m=size(indclass2,1)*2;
else
    indclass2=indclass2(1:size(indclass1,1),:);
    m=size(indclass1,1)*2;
end

imagen1=obj.clase(indclass1,1);
objeto1=obj.clase(indclass1,2);
clase1=obj.clase(indclass1,3);
imagen2=obj.clase(indclass2,1);
```

```
objeto2=obj.clase(indclass2,2);
clase2=obj.clase(indclass2,3);

imagen3=[imagen1; imagen2];
objeto3=[objeto1; objeto2];
clase3=[clase1; clase2];


for k=1:objTest.obj(indTest,2)

    ObjTest=load([Test sprintf('%d', imagenTest) '_'
sprintf('%d',id_obj(k,1)) '.mat']);
    descTest=ObjTest.descriptors(:,:);
    n=size(descTest,1);



    distmas=zeros(m,n);
    distmenos=zeros(m,n);
    for i=1:m

        imagen=imagen3(i,1);
        objeto=objeto3(i,1);
        clase=clase3(i,1);

        Train=load([TrainRec sprintf('%d', imagen) '_' sprintf('%d',
objeto) '.mat']); % imagen solo con el objeto

        descTrain=Train.descriptors;

        for j=1:n
            aux = repmat(descTest(j,:),size(descTrain,1),1)-descTrain;
            dist_desc= sum(aux.*aux,2);
            val=min(dist_desc);

            if (clase==1)
                distmas(i,j)=val;
            else
                distmenos(i,j)=val;
            end
        end

    end



    %% Me quedo con la min distancia de cada punto de la imagen de Test a
el
    %% objeto correspondiente
    distmenos=distmenos';
    distmas=distmas';
    distanciamenos=zeros(1,size(distmenos,1));
    distanciamas=zeros(1,size(distmas,1));
    for i=1:size(distmenos,1)
        aux=nonzeros(distmenos(i,:));
        if(isempty(aux))
            distanciamenos(i)=1;
        else
```

```
            distanciamenos(i)=min(aux);
        end
    end
    for i=1:size(distmas,1)
        aux=nonzeros(distmas(i,:));
        if(isempty(aux))
            distanciamas(i)=1;
        else
            distanciamas(i)=min(aux);
        end
    end



    dist_Promedio=zeros(size(descTest,1),3);
    for j=1:size(descTest,1)
        dist_Promedio(j,1)=j;
        dist_Promedio(j,2)=distanciamenos(j);
        dist_Promedio(j,3)=distanciamas(j);
    end



    file=([distancia 'distTest' sprintf('%d', imagenTest) '_'
sprintf('%d',id_obj(k,1)) '.mat']);
    save(file,'dist_Promedio');

end

%% Cogemos el doble de imagenes de train de clase 2, con mas puntos de
%% clase 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion clase2                                                    %%
%%% Argumento de entrada: Nombre de la imagen                         %%
%%% Salida: Distancias de test a train                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function clase2(imagenTest)

addpath([cd '/Main/']);
distancia=([cd '/DistancesC2/']);
TrainRec=([cd '/TrainRec/']);
Test=([cd '/TestRec/']);



objTest=load([Test 'ClaseRealTest.mat']);
indTest=find(objTest.obj(:,1)==imagenTest);
ind=find(objTest.clase(:,1)==imagenTest);
id_obj=objTest.clase(ind,2);

obj=load([TrainRec 'ClaseRealTrain.mat']);
m=size(obj.clase,1);



for k=1:objTest.obj(indTest,2)
```

```
    ObjTest=load([Test sprintf('%d', imagenTest) '_'
sprintf('%d',id_obj(k,1)) '.mat']);
    descTest=ObjTest.descriptors(:,:);
    n=size(descTest,1);


    distmas=zeros(m,n);
    distmenos=zeros(m,n);
    for i=1:m


        imagen=obj.clase(i,1);
        objeto=obj.clase(i,2);
        clase=obj.clase(i,3);
        Train=load([TrainRec sprintf('%d', imagen) '_' sprintf('%d',
objeto) '.mat']); % imagen solo con el objeto

        descTrain=Train.descriptors;
        for j=1:n
            aux = repmat(descTest(j,:),size(descTrain,1),1)-descTrain;
            dist_desc= sum(aux.*aux,2);
            val=min(dist_desc);

            if (clase==1)
                distmas(i,j)=val;
            else
                distmenos(i,j)=val;
            end
        end

    end


    %% Me quedo con la min distancia de cada punto de la imagen de Test a
el
    %% objeto correspondiente
    distmenos=distmenos';
    distmas=distmas';
    distanciamenos=zeros(1,size(distmenos,1));
    distanciamas=zeros(1,size(distmas,1));
    for i=1:size(distmenos,1)
        aux=nonzeros(distmenos(i,:));
        if(isempty(aux))
            distanciamenos(i)=1;
        else
            distanciamenos(i)=min(aux);
        end
    end
    for i=1:size(distmas,1)
        aux=nonzeros(distmas(i,:));
        if(isempty(aux))
            distanciamas(i)=1;
        else
            distanciamas(i)=min(aux);
        end
    end
```

```
    dist_Promedio=zeros(size(descTest,1),3);
    for j=1:size(descTest,1)
        dist_Promedio(j,1)=j;
        dist_Promedio(j,2)=distanciamenos(j);
        dist_Promedio(j,3)=distanciamas(j);
    end



    file=([distancia 'distTest' sprintf('%d', imagenTest) '_'
sprintf('%d',id_obj(k,1)) '.mat']);
    save(file,'dist_Promedio');

end

%% Cogemos el mismo numero de puntos en las dos clases
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion eqpnts                                                    %%
%%% Argumento de entrada: Nombre de la imagen                         %%
%%% Salida: Distancias de test a train                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function eqpnts(imagenTest)

addpath([cd '/Main/']);
distancia=([cd '/DistancesEq/']);
TrainRec=([cd '/TrainRec/']);
Test=([cd '/TestRec/']);


objTest=load([Test 'ClaseRealTest.mat']);
indTest=find(objTest.obj(:,1)==imagenTest);
ind=find(objTest.clase(:,1)==imagenTest);
id_obj=objTest.clase(ind,2);

obj=load([TrainRec 'ClaseRealTrain.mat']);


indclass1=find(obj.clase(:,3)==1);
indclass2=find(obj.clase(:,3)==-1);

if(size(indclass1,1)>size(indclass2,1))
    indclass1=indclass1(1:size(indclass2,1),:);
    m=size(indclass2,1)*2;
else
    indclass2=indclass2(1:size(indclass1,1),:);
    m=size(indclass1,1)*2;
end

imagen1=obj.clase(indclass1,1);
objeto1=obj.clase(indclass1,2);
clase1=obj.clase(indclass1,3);
imagen2=obj.clase(indclass2,1);
objeto2=obj.clase(indclass2,2);
```

```
clase2=obj.clase(indclass2,3);

imagen3=[imagen1; imagen2];
objeto3=[objeto1; objeto2];
clase3=[clase1; clase2];


for k=1:objTest.obj(indTest,2)

    ObjTest=load([Test sprintf('%d', imagenTest) '_'
sprintf('%d',id_obj(k,1)) '.mat']);
    descTest=ObjTest.descriptors(:,:);
    n=size(descTest,1);



    distmas=zeros(m,n);
    distmenos=zeros(m,n);
    for i=1:m

        imagen=imagen3(i,1);
        objeto=objeto3(i,1);
        clase=clase3(i,1);

        Train=load([TrainRec sprintf('%d', imagen) '_' sprintf('%d',
objeto) '.mat']); % imagen solo con el objeto

        descTrain=Train.descriptors;

        for j=1:n
            aux = repmat(descTest(j,:),size(descTrain,1),1)-descTrain;
            dist_desc= sum(aux.*aux,2);
            val=min(dist_desc);

            if (clase==1)
                distmas(i,j)=val;
            else
                distmenos(i,j)=val;
            end
        end

    end


    %% Me quedo con la min distancia de cada punto de la imagen de Test a
el
    %% objeto correspondiente
    distmenos=distmenos';
    distmas=distmas';
    distanciamenos=zeros(1,size(distmenos,1));
    distanciamas=zeros(1,size(distmas,1));
    for i=1:size(distmenos,1)
        aux=nonzeros(distmenos(i,:));
        if(isempty(aux))
            distanciamenos(i)=1;
        else
            distanciamenos(i)=min(aux);
```

```
        end
    end
    for i=1:size(distmas,1)
        aux=nonzeros(distmas(i,:));
        if(isempty(aux))
            distanciamas(i)=1;
        else
            distanciamas(i)=min(aux);
        end
    end



    dist_Promedio=zeros(size(descTest,1),3);
    for j=1:size(descTest,1)
        dist_Promedio(j,1)=j;
        dist_Promedio(j,2)=distanciamenos(j);
        dist_Promedio(j,3)=distanciamas(j);
    end



    file=([distancia 'distTest' sprintf('%d', imagenTest) '_'
sprintf('%d',id_obj(k,1)) '.mat']);
    save(file,'dist_Promedio');

end
```

BOW

```
%% Funcion que separa los archivos de Train en 2 grupos, el 60% para
%% realizar kmeans y al resto se le aplicaran los clusters y se entrenara
%% el SVM
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;

% Añadimos carpetas necesarias y cargamos archivo con nombre de imagenes
de
% train
main=([cd '/Main/']);
pntsTrain=[cd '/SiftTrain/'];
archivo=load([main 'car_train.txt']);
clase0=find(archivo(:,2)==0);
archivo(clase0,2)=-1;

percentTrain=round(2501*0.6); %%cogemos el 60% de los coches para hacer
el kmeans
contador=1;
```

```
matriz=[];
imagenes=0;


rp=randperm(2501); % variable para desordenar las iamgenes de train

archivo1=archivo(rp(1:percentTrain),:); % archivo 1 con las ids y clases
con las que se realizara el kmeans
archivo2=archivo(rp(percentTrain+1:size(archivo,1)),:); % archivo 2 con
las ids y clases con las que se entrenera el SVM

file='car_train_div.mat';
save(file,'archivo1','archivo2'); % guardamos ambos archivos


ind1=find(archivo1(:,2)==1);
clase1=size(ind1,1);

ind2=find(archivo1(:,2)==-1);
clase2=size(ind2,1);
% igualamos clases
if(clase1>clase2)
    claset=clase2;
else
    claset=clase1;
end

% igualamos clases 50% clase 1 y 50% clase -1 pero solo en imagenes
for i=1:claset%% Solo me quedo con  1501 (60%) de las imagenes para hacer
el clustering
    puntos=load([pntsTrain sprintf('%d',archivo1(ind1(i),1)) '.mat']);
    matriz(contador:contador+size(puntos.descriptors,1)-
1,:)=puntos.descriptors;
    contador=contador+size(puntos.descriptors,1);
end
for i=1:claset%% Solo me quedo con  1501 (60%) de las imagenes para hacer
el clustering
    puntos=load([pntsTrain sprintf('%d',archivo1(ind2(i),1)) '.mat']);
    matriz(contador:contador+size(puntos.descriptors,1)-
1,:)=puntos.descriptors;
    contador=contador+size(puntos.descriptors,1);
end

file='matriz.mat';
save(file,'matriz','rp');

clustering(); % funcion que realiza el kmeans y que luego aplica los
clusters al resto de imagenes de train
SupportVector(); % funcion que entrena el SVM y que aplica clusters a
test y clasifica mediante SVM


%% Funcion donde se ejecuta kmeans y donde se asignan los puntos
restantes
%% a los clusters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion Clustering                                                 %%
```

```
%%% Salida: Archivos con los puntos de entrenamiento      %%
%%%         asignados a los clusters                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function clustering()

addpath([cd '/Main/']);
pntsTrain=[cd '/SiftTrain/'];

matriz=load('matriz.mat');

k=[250,500,1000,2000]; %%De 1000 clusters
[Cmin, dmin]=km2(k(1),matriz);

%% realizamos el kmeans
for i=1:length(k)
    for j=1:10
        [C, dtotal]=km2(k(i),matriz);
        if(dtotal < dmin)
            dmin=dtotal;
            Cmin=C;
        end

    end
    file=(['clusters' sprintf('%d', k(i)) '.mat']);
    save(file,'Cmin', 'dmin');
 end


%% Describimos  el resto de las  imagenes de train en base a los clusters
que hemos
%%definido antes

archivo=load('car_train_div.mat');
archivo2=archivo.archivo2;

train2=size(archivo2,1);
 c=[250,500,1000,2000];
for h=1:length(c)

    clusters=load(['clusters' sprintf('%d',c(h)) '.mat']);
    pathvTrain=([cd '/vTrain_' sprintf('%d',c(h)) '/']);
    for i=1:train2
        puntos=load([pntsTrain sprintf('%d',archivo2(i,1)) '.mat']);
        cluster = zeros(1,size(puntos.descriptors,1));

        %por min dist elijo que cluster es al que pertenece cada punto
        for j=1:size(puntos.descriptors,1)
            aux =
repmat(puntos.descriptors(j,:),size(clusters.Cmin,1),1)-clusters.Cmin;
            dist_centroide= sum(aux.*aux,2);
            [val,indx]=min(dist_centroide);
            cluster(j)= indx;
        end

        vector = zeros(1,size(clusters.Cmin,1));
        % media de la cantidad de puntos que pertencen a un cluster con
```

```
        % respecto a los putnos totales de la imagen
        for k=1:size(clusters.Cmin,1)

vector(k)=size(find(cluster==k),2)/size(puntos.descriptors,1);
        end
        file=[pathvTrain 'vTrain' sprintf('%d', archivo2(i,1)) '.mat'];
        save(file,'vector');

    end

    % paso todos los vectors a una matriz y la guardo y su clase
    training=zeros(train2,c(h));
    for i=1:train2
        vector=load([pathvTrain 'vTrain' sprintf('%d', archivo2(i,1))
'.mat']);
        training(i,:)=vector.vector;
    end
    clase=archivo2(:,2);
    file=['SVMtrain' sprintf('%d',c(h)) '.mat'];
    save(file,'training','clase');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion Km2                                                       %%
%%% Argumento de entrada: numero de clusters                          %%
%%%                       datos                                       %%
%%% Salida: clusters, distancia minima                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [C,dtotal]=km2(nc,puntos)


%nc --> numero de centros/cluster
npatsClust = 100000; % numero de puntos para hacer el clutering
npatsIncr = 10000;

x=puntos.matriz;
[n, d] = size(x);

tic;
options = statset('Display','iter');
rp = randperm(n);
[ix C] =
kmeans(x(rp(1:npatsClust),:),nc,'EmptyAction','singleton','options',optio
ns,'onlinephase','off');
toc

d = pdist2(x(rp(1:npatsClust),:),C);
[dd ix2] = min(d,[],2);
sum(dd.*dd)

tic;
ini = 1;
fin = npatsIncr;
dtotal = 0;
while (true)
    xaux = x(ini:fin,:);
```

```
    d = pdist2(xaux,C);
    [dd ix2] = min(d,[],2);
    fprintf('%6d  %6d  %f\n',ini,fin,sum(dd.*dd));
    dtotal = dtotal + sum(dd.*dd);
    if fin == n
        break;
    end
    ini = fin + 1;
    fin = min(n,fin + npatsIncr);
end
toc

fprintf('dtotal = %f\n',dtotal);



%% Funcion que entrena SVM y predice con las iamgenes de Test
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion SupportVector                                                 %%
%%% Salida: Resultados del clasificador SVM                               %%
%%%                                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function SupportVector()

clear all;
close all;
clc;



addpath([cd '/Main/']);
pntsTest=[cd '/SiftTest/'];
percentTest=2510; %%el 40% de los coches

%% A continuación predecimos con el SVM las clases de las imagenes de
test
archivo2=load('car_val.txt');
clase0=find(archivo2(:,2)==0);
archivo2(clase0,2)=-1;
claseTest=archivo2(:,2);

 c=[250,500,1000,2000];
for h=1:length(c)
    pathvector=([cd '/vector_' sprintf('%d',c(h)) '/']);
    clusters=load(['clusters' sprintf('%d',c(h)) '.mat']);
    for i=1:percentTest
        puntos=load([pntsTest sprintf('%d', archivo2(i,1)) '.mat']);
        cluster = zeros(1,size(puntos.descriptors,1));
        for j=1:size(puntos.descriptors,1)
            aux =
repmat(puntos.descriptors(j,:),size(clusters.Cmin,1),1)-clusters.Cmin;
            dist_centroide= sum(aux.*aux,2);
            [val,indx]=min(dist_centroide);
            cluster(j)= indx;
        end
        % media de la cantidad de puntos que pertencen a un cluster con
        % respecto a los putnos totales de la imagen
```

```
        vector = zeros(1,size(clusters.Cmin,1));
        for k=1:size(clusters.Cmin,1)

vector(k)=size(find(cluster==k),2)/size(puntos.descriptors,1);
        end
        file=[pathvector 'vector' sprintf('%d',archivo2(i,1)) '.mat'];
        save(file,'vector');
    end
    % paso todos los vectors a una matriz y la guardo y su clase
    testing=zeros(percentTest,c(h));
    for i=1:percentTest
        vector=load([pathvector 'vector' sprintf('%d', archivo2(i,1))
'.mat']);
        testing(i,:)=vector.vector;
    end
    clase=claseTest;
    file=['SVMtest' sprintf('%d',c(h)) '.mat'];
    save(file,'testing','clase');



end
  %% Entreno utilizando SVM
  SVMtrain=load('SVMtrain.mat');
  model=svmtrain(SVMtrain.clase,SVMtrain.training,'-t 0 -c 15 -b 1');
  file='modelSVM.mat';
  save(file,'model');



%% Clasifico utilizando SVM
SVMtest=load('SVMtest.mat');
SVMtrain=load('SVMtrain.mat');
modelSVM=load('modelSVM.mat');


[predicted_label, accuracy, prob_estimates] = svmpredict(SVMtest.clase,
SVMtest.testing, modelSVM.model,'-b 1');


file=('prob_estimatesBOW.mat');
save(file,'prob_estimates','predicted_label', 'accuracy');
```

En el caso de que se vaya a aplicar SIFT directamente en la imagen:

```
% [image, descriptors, locs] = sift(imageFile)
%
% This function reads an image and returns its SIFT keypoints.
%   Input parameters:
%     imageFile: the file name for the image.
%
%   Returned:
%     image: the image array in double format
%     descriptors: a K-by-128 matrix, where each row gives an invariant
%         descriptor for one of the K keypoints.  The descriptor is a
vector
```

```
%         of 128 values normalized to unit length.
%     locs: K-by-4 matrix, in which each row has the 4 values for a
%         keypoint location (row, column, scale, orientation).  The
%         orientation is in the range [-PI, PI] radians.
%
% Credits: Thanks for initial version of this program to D. Alvaro and
%          J.J. Guerrero, Universidad de Zaragoza (modified by D. Lowe)

function [image, descriptors, locs] = sift(imageFile)

% Load image
image = imread(imageFile);

% If you have the Image Processing Toolbox, you can uncomment the
following
%   lines to allow input of color images, which will be converted to
grayscale.
% if isrgb(image)
%    image = rgb2gray(image);
% end

[rows, cols] = size(image);

% Convert into PGM imagefile, readable by "keypoints" executable
f = fopen('tmp.pgm', 'w');
if f == -1
    error('Could not create file tmp.pgm.');
end
fprintf(f, 'P5\n%d\n%d\n255\n', cols, rows);
fwrite(f, image', 'uint8');
fclose(f);

% Call keypoints executable
if isunix
    command = '!./sift ';
else
    command = '!siftWin32 ';
end
command = [command ' <tmp.pgm >tmp.key'];
eval(command);

% Open tmp.key and check its header
g = fopen('tmp.key', 'r');
if g == -1
    error('Could not open file tmp.key.');
end
[header, count] = fscanf(g, '%d %d', [1 2]);
if count ~= 2
    error('Invalid keypoint file beginning.');
end
num = header(1);
len = header(2);
if len ~= 128
    error('Keypoint descriptor length invalid (should be 128).');
end

% Creates the two output matrices (use known size for efficiency)
```

```
locs = double(zeros(num, 4));
descriptors = double(zeros(num, 128));

% Parse tmp.key
for i = 1:num
    [vector, count] = fscanf(g, '%f %f %f %f', [1 4]); %row col scale ori
    if count ~= 4
        error('Invalid keypoint file format');
    end
    locs(i, :) = vector(1, :);

    [descrip, count] = fscanf(g, '%d', [1 len]);
    if (count ~= 128)
        error('Invalid keypoint file value.');
    end
    % Normalize each input vector to unit length
    descrip = descrip / sqrt(sum(descrip.^2));
    descriptors(i, :) = descrip(1, :);
end
fclose(g);
```

En el caso de que se vaya a aplicar SIFT después del mallado de la imagen:

```
%% Funcion principal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion Principal que llama al resto de las funciones %%
%%% Las imagenes se escalan, se mallan, y se les aplica   %%
%%% SIFT                                                  %%
%%%                                                       %%
%%%                                                       %%
%%%                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

% Carpetas de entrada
main=[cd '/Main/'];
train=[cd '/ImagTrain/'];
test=[cd '/ImagTest/'];
pntstrain=[cd '/pntsTrain/'];
pntstest=[cd '/pntsTest/'];
pntstrain2=[cd '/pntsTrain_1000/'];
pntstest2=[cd '/pntsTest_1000/'];
HarrisTrain=[cd '/HarrisTrain/'];
HarrisTest=[cd '/HarrisTest/'];
HarrisTrain2=[cd '/HarrisTrain_1000/'];
HarrisTest2=[cd '/HarrisTest_1000/'];
sifthtrain=[cd '/SiftHTrain/'];
sifthtest=[cd '/SiftHTest/'];
sifthtrain2=[cd '/SiftHTrain_1000/'];
sifthtest2=[cd '/SiftHTest_1000/'];

%Cargamos archivos
archivo=load([main 'car_train.txt']);
archivo2=load([main 'car_val.txt']);
```

```
archivo0=find(archivo(:,2)==0);
archivo(archivo0,2)=-1;
archivo0=find(archivo2(:,2)==0);
archivo2(archivo0,2)=-1;

[m n]=size(archivo);
[m2 n2]=size(archivo2);

% Escalamos las imagenes
pixel=100000;
escala(pixel);

% Mallamos las imagenes y nos quedamos con la posicion
for i=1:m
   disp(i)
   mallar(archivo(i,1),train,pntstrain);
end
  disp('Mallado Test\n')
for i=1:m2
   disp(i)
   mallar(archivo2(i,1),test,pntstest);
end

% Creamos ficheros con las circunf
disp('Circunf Train\n')

for i=1:m
   disp(i)
   k=0.01;
   harris(archivo(i,1),pntstrain,HarrisTrain,k);
end
 disp('Circunf Test\n')
for i=1:m2
   disp(i)
   k=0.01;
   harris(archivo2(i,1),pntstest,HarrisTest,k);
end

% Aplicamos sift a los archivos
disp('Sift para Train\n')
for i=1:m
    disp(i)
   descript(archivo(i,1),train,HarrisTrain,sifthtrain)
end
disp('Sift para Test\n')
for i=1:m2
   disp(i)
   descript(archivo2(i,1),test,HarrisTest,sifthtest)
end



%% Funcion para escalar imagenes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion Escala                                                        %%
%%% Argumento de entrada: Numero de pixeles                               %%
```

```
%%%                         en la iamgen a escalar          %%
%%%                                                         %%
%%% Salida: Imagenes escaladas                              %%
%%%                                                         %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function escala(pixel)
% Carpetas
main=[cd '/Main/'];
trainin=[cd '/Car_Train_PGM/'];
testin=[cd '/Car_Val_PGM/'];
trainout=[cd '/ImagTrain/'];
testout=[cd '/ImagTest/'];

%Cargamos archivos
archivo=load([main 'car_train.txt']);
archivo2=load([main 'car_val.txt']);

archivo0=find(archivo(:,2)==0);
archivo(archivo0,2)=-1;
archivo0=find(archivo2(:,2)==0);
archivo2(archivo0,2)=-1;

[m n]=size(archivo);
[m2 n2]=size(archivo2);

%Escalamos todas las iamgenes
for i=1:m
   a=length(num2str(archivo(i,1)));
    if(a<6)
        path=([sprintf('%d',zeros(1,6-a)) sprintf('%d',archivo(i,1))]);
    else
        path=sprintf('%d',archivo(i,1));
    end
    I = imread([trainin path],'pgm');
    [h w]=size(I);
    k=sqrt(100000/(h*w));
    h1=k*h;
    w1=k*w;
    J = imresize(I, [h1 w1]);
    imwrite(J,[trainout path '.pgm']);
end

for i=1:m2
   a=length(num2str(archivo2(i,1)));
    if(a<6)
        path=([sprintf('%d',zeros(1,6-a)) sprintf('%d',archivo2(i,1))]);
    else
        path=sprintf('%d',archivo(i,1));
    end
    I = imread([testin path '.pgm']);
    [h w]=size(I);
    k=sqrt(pixel/(h*w));
    h1=k*h;
    w1=k*w;
    J = imresize(I, [h1 w1]);
    disp(i)
    imwrite(J,[testout path '.pgm']);
end
```

```
%% funcion de mallado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion Mallado                                                    %%
%%% Argumento de entrada: Nombre de la imagen                          %%
%%%                       carpeta de entrada                           %%
%%%                       carpeta de salida                            %%
%%% Salida: Archivo con las posiciones del mallado                     %%
%%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mallar(imagen,path,pathout)
a=length(num2str(imagen));
if(a<6)
    im=([sprintf('%d',zeros(1,6-a)) sprintf('%d',imagen)]);
else
    im=sprintf('%d',imagen);
end
matriz=imread([path im], 'pgm');

[h w]=size(matriz);

dx=round(w/10); % para 1000 pnts /15
dy=round(h/10);
puntos=zeros(250,2);
f=1;
for y=1:dy:h
    for x=1:dx:w
       if((x<=w) && (y<=h))
           puntos(f,1)=x;
           puntos(f,2)=y;
           f=f+1;
       end

    end
end
file=([pathout sprintf('%d',imagen) '.mat']);
save(file,'puntos')



% Funcion para crear circunf sobre los puntos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion Harris                                                     %%
%%% Argumento de entrada: Nombre de la imagen                          %%
%%%                       carpeta de entrada                           %%
%%%                       carpeta de salida                            %%
%%% Salida: Archivo con las regiones de las                            %%
%%%         circunferencias                                            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function harris(imagen,path,pathout,k)

locs=load([path sprintf('%d',imagen) '.mat']);

ind0=find(locs.puntos(:,1)==0);
if(~isempty(ind0))
    loc=locs.puntos(1:ind0(1)-1,:);
else
```

```
    loc=locs.puntos;
end
circunf=5;
m=size(loc,1);% numero de puntos en cada escala
%valores de las circunf
a=k; % valor del radio de la circunf
b=0;
f = fopen([pathout sprintf('%d',imagen) '.txt'], 'wt');
if f == -1
    error('Could not create file');
end
fprintf(f, '1.0\n'); %% dimension del vector
fprintf(f, '%d\n', m*circunf); % numero de puntos caracteristicos
for c=1:circunf

    for i=1:m
        fprintf(f, '%d %d %d %d %d\n',loc(i,1), loc(i,2),a,b,a);
    end

    a=a/2;
end
 fclose(f);

% funcion con la que vamos a aplicar SIFT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion Descript                                                      %%
%%% Argumento de entrada: Nombre de la imagen                             %%
%%%                       carpeta de entrada de la imagen  %%
%%%                       carpeta de entrada con los textos%%
%%%                       carpeta de salida                %%
%%% Salida: Archivo con los descriptores                                  %%
%%%                                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function descript(imagen,path,pathtxt,pathout)

if isunix
    command = '!./compute_descriptors.ln';
else
     error('Could not execute the program');
end
a=length(num2str(imagen));
if(a<6)
    path1=([sprintf('%d',zeros(1,6-a)) sprintf('%d',imagen)]);
else
    path1=sprintf('%d',imagen);
end
path2=[path path1];
path2txt=[pathtxt sprintf('%d',imagen)];
path2out=[pathout sprintf('%d',imagen)];
command = [command ' -sift -i ' path2 '.pgm -p1 ' path2txt '.txt -o1 '
path2out '.txt'];
eval(command);
```

Funciones para dibujar las curvas de Precision/Recall y ROC

```
%% Funcion para claculo de las Curvas Precision/Recall y ROC
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funcion savedatos                                                   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;
NBNN=[cd '/NBNN/'];
ROC=[cd '/ROCNBNN/'];
fig=[cd '/figuras/'];
Imag=[cd '/JPEGImages/'];


for i=1:3

    dist=load([NBNN 'Matriz_distPromedioClase' sprintf('%d',i) '.mat']);
if(i==3)
        orden='descend';
    else
        orden='ascend';
    end
    [vals1,indx1] = sort((real(dist.Matriz_distPromedio(:,4))),orden);

if(i==3)
        vals1=vals1;
    else
        vals1=-vals1;
    end

    archivo=load([NBNN 'car_val.txt']);
    clase0=find(archivo(:,2)==0);
    archivo(clase0,2)=-1;
    clasereal=archivo(:,2);
    NGrupo=1:1:size(clasereal,1);
    NTotal=size(find(clasereal==1),1);
    ClaseOrdenada=clasereal(indx1);


    Precision=zeros(1,length(NGrupo));
    Recall=zeros(1,length(NGrupo));
    TVP=zeros(1,length(NGrupo));
    TFP=zeros(1,length(NGrupo));
    for j=1:length(NGrupo)

        N1Grupo=size(find(ClaseOrdenada(1:NGrupo(j))==1),1);
        Precision(j)=N1Grupo/NGrupo(j);
        Recall(j)=N1Grupo/NTotal;
        TVP(j) = Recall(j);
        TFP(j) = (NGrupo(j)-N1Grupo)/(2510-NTotal);

    end

    [X,Y,THRE,AUC,OPTROCPT,SUBY,SUBYNAMES] =
perfcurve(ClaseOrdenada,vals1,1);
    file=[ROC 'NBNNPC' sprintf('%d',i) '.mat'];
    save(file,'Recall','Precision','AUC');
```

```
    figure(1)
    plot([0 1],[1 0],'r',Recall,Precision,'b');
    title('Cars')
    xlabel('Recall');
    ylabel('Precision');
    axis([0 1 0 1]);
    grid on;
    saveas(1,[fig 'NBNNPC' sprintf('%d',i) '.jpg' ])


    figure(2)
    plot(X,Y,'r',TFP,TVP)
```

```
%Funcion para crear grafica de AUC frente a clusters
clear all;
close all;
clc;
ROC=[cd '/ROC/'];

c=[100 200 300 500 700];
AUC1=zeros(1,length(c));
for i=1:length(c)
    archivo=load([ROC 'BOW' sprintf('%d',c(i)) '.mat']);
    AUC1(i)=archivo.AUC;
end


figure
plot(c,AUC1,'s-')
title('AUC vs Cluster')
xlabel('Numero de Clusters');
ylabel('AUC');
grid on;
```

Función para reducir la dimensión : PCA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Funcion para aplicar reduccion de dimensionalidad
%
% Utilizando PCA
%
% Argumentos de entrada: idir : carpeta con los datos de entrada
%
%                        odir : carpeta de salida
%
%                        numcomp: variable que define el tamaño a reducir
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
```

```
function computePCA(idir,odir,numcomp)


aux = dir(idir);
aux = aux(3:end);
nfiles = length(aux);

% Primera pasada para calcular la media:
media = zeros(1,128);
%data = [];
numcasos = 0;
for i = 1:nfiles

    fname = [idir '/' aux(i).name];
    if mod(i,100) == 0
        fprintf('Pasada 1: ... processing file %d of %d
<%s>\n',i,nfiles,fname);
    end
    load(fname);

    media = media + sum(points.sift);
    numcasos = numcasos + size(points.sift,1);
    %data = [data; points.sift];
end

media = media / numcasos;

% Segunda pasada para calcular la std de cada atributo:
s2 = zeros(1,128);
for i = 1:nfiles

    fname = [idir '/' aux(i).name];
    if mod(i,100) == 0
        fprintf('Pasada 2: ... processing file %d of %d
<%s>\n',i,nfiles,fname);
    end
    load(fname);

    paux = points.sift - repmat(media,size(points.sift,1),1);
    s2 = s2 + sum(paux.*paux);
end

s2 = s2 / (numcasos-1);
s = sqrt(s2);

% Tercera pasada para calcular la matriz de covarianzas y hacer PCA:
cm = zeros(128,128);
for i = 1:nfiles

    fname = [idir '/' aux(i).name];
    if mod(i,100) == 0
        fprintf('Pasada 3: ... processing file %d of %d
<%s>\n',i,nfiles,fname);
    end
    load(fname);
```

```
    % Estandarizo:
    paux = (points.sift - repmat(media,size(points.sift,1),1)) ./
repmat(s,size(points.sift,1),1);
    cm = cm + paux'*paux;
end

cm = cm / (numcasos-1);


% PCA:
[v d] = eig(cm);
d = diag(d);
[d ix] = sort(d,'descend');
d = d(1:numcomp);
v = v(:,ix(1:numcomp));

% Cuarta pasada para aplicar PCA a cada fichero:
for i = 1:nfiles

    fname = [idir '/' aux(i).name];
    if mod(i,100) == 0
        fprintf('Pasada 4: ... processing file %d of %d
<%s>\n',i,nfiles,fname);
    end
    load(fname);
    pointsaux = points;
    points = [];
    points.pos = pointsaux.pos;

    pdata = pointsaux.sift*v;
    points.sift = pdata;
    foutname = sprintf('%s/%s.pca%03d.mat',odir,aux(i).name(1:(end-
4)),numcomp);
    save(foutname,'points','-mat');

end
```

PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... ..... 2.000 €
- Alquiler de impresora láser durante 9 meses ..... 75 €
- Material de oficina ..... 150 €
- Total de ejecución material..... 2.225 €

2) Gastos generales

- 16 % sobre Ejecución Material..... 356 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material..... 133.5 €

4) Honorarios Proyecto

- 1000 horas a 15 € / hora..... 15000 €

5) Material fungible

- Gastos de impresión ..... 100 €
- Encuadernación ..... 20 €

6) Subtotal del presupuesto

- Subtotal Presupuesto..... 17834.5 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto..... 3745,24 €

8) Total presupuesto

- Total Presupuesto..... 21579.74 €

Madrid, Septiembre de 2012

Ingeniero Jefe de Proyecto

Fdo. : BADER AUDEH PIÑAR

Ingeniero de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un sistema de detección automática de objetos. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de

obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.