Visión Artificial

Práctica 3

Azael Escudero López y Sergio Ferrer Gimeno (GII+GIS)



Tabla de contenido

Algoritmo Desarrollado 3

Parte 1: Detección del centro del coche con ORB y votación a la Hough 3

Parte 2: Detección de coches utilizando HAAR 6

Parte 3: Detección de coche en secuencias de vídeo 7

Estadísticas 7

Conclusiones 8

# Algoritmo Desarrollado

El algoritmo desarrollado en la práctica 3 tiene como finalidad última localizar los coches que hay en una imagen, primero utilizando el objeto ORB de OPENCV que ejecutará los algoritmos FAST y BRIEF de visión artificial para encontrar los KeyPoints (esquinas) de la imagen y los descritores de los mismos, y luego utilizando una votación a la HOUGH con ayuda de un índice flan.  
 Posteriormente se utilizará otro método de detección de coches a través de HAAR y se aplicará tanto en imágenes como en vídeos. Este último sistema es el que, posteriormente, se aplicará en la práctica 4.

## Parte 1: Detección del centro del coche con ORB y votación a la Hough

Para que este sistema funciona lo primero que ha de hacerse es un entrenamiento que le permitirá reconocer posteriormente los coches en las imágenes de test.

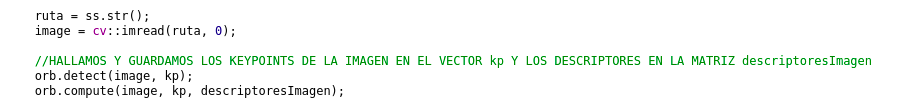
Este entrenamiento se hará en este caso con 48 imágenes de training en las que los coches aparecerán perfectamente centrados y con los bordes recortados alrededor de los mismos de tal manera que se le pueda decir al sistema dónde está el centro del coche aproximadamente sin realmente saberlo (el centro del coche estará en el centro de la imagen).



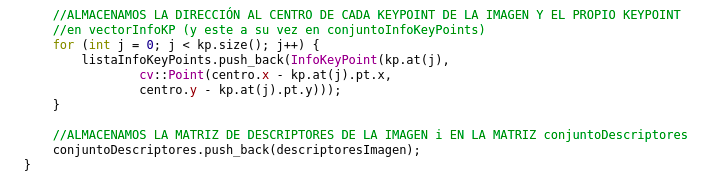
Ejemplo Imagen de Training

Para realizar este entrenamiento se detectarán todos los KeyPoints de las imágenes y sus descriptores utilizando ORB y se irán guardando en una estructura de datos que más tarde se pasará al constructor del índice flann. La creación del índice marca el fin de la parte de entrenamiento del sistema.

Como sabemos la posición de cada KeyPoint en la imagen, también deberemos guardar el cálculo del vector que une el keyPoint con el centro del coche (de la imagen), ya que será lo que posteriormente nos permitirá hacer una estimación del lugar donde se encuentra el coche en las imágenes de test mediante la votación a la Hough.



Detección de KeyPoints y de Descriptores



Cálculo del Vector al Centro



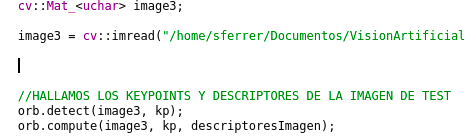
Creación del Índice Flann

Una vez entrenado el sistema se puede comenzar a probar su funcionamiento con imágenes reales en las que los coches no aparecen perfectamente encuadrados ni en posición frontal.

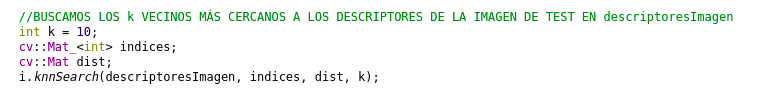
Para ello realizaremos el mismo proceso que con las imágenes de entrenamiento pero con algún cambio, es decir, detectaremos los KeyPoints y descriptores (el vector al centro no porque aquí no sabemos donde está el coche) y le pasaremos los descriptores al índice flann pero en vez de para crearlo, para que los compare (mediante el método knnSearch) con los que le pasamos al constructor, es decir, los de las imágenes de entrenamiento.

Mediante la distancia de Hamming los comparará y nos dará una lista de los índices a los k vecinos con descriptores más parecidos al descriptor que le hemos pasado. Con estos índices realizaremos la votación a la Hough.

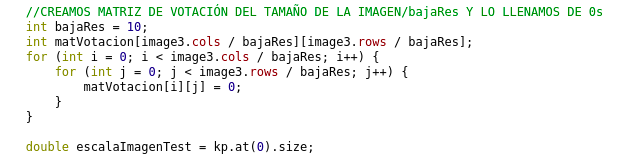
Para realizar esta votación se recorrerá la lista de índices accediendo a los KeyPoints que corresponde a cada índice y a su vector al centro, se calculará el supuesto centro al que cada KeyPoint apunta y se realizará la votación, el punto que reciba más votos será el centro del coche.



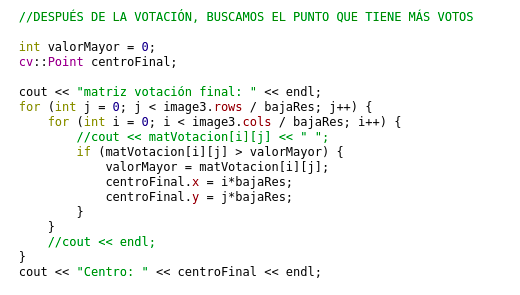
Detección de KeyPoints y de Descriptores



Búsqueda de los k vecinos



Inicialización de la Matriz de Votos

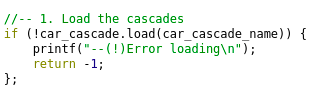


Búsqueda del punto con más votos

Una vez hallado el centro se dibuja un rectángulo alrededor del coche y el coche quedaría detectado.

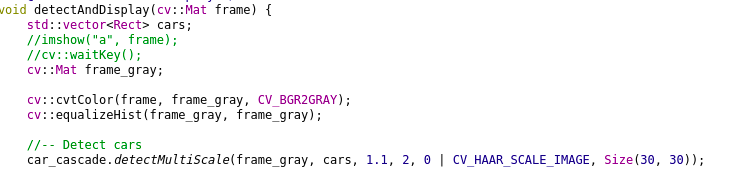
## Parte 2: Detección de coches utilizando HAAR

Esta parte es más sencilla y corta. Consiste en lo mismo que lo anterior pero utilizando HAAR en vez de ORB y la votación a la Hough. Para realizarla se cargará en un objeto cv::CascadeClassifier que hemos llamado car\_cascade un archivo xml con la información de entrenamiento de un sistema HAAR.



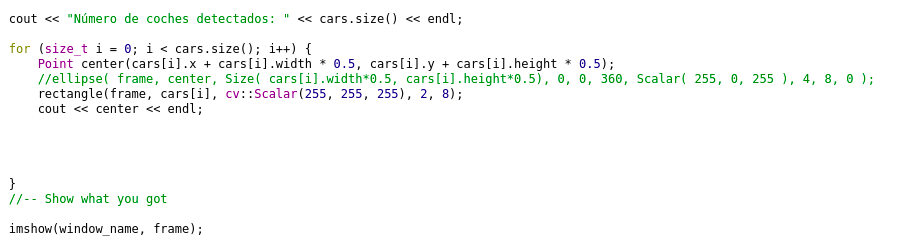
Carga del CascadeClassifier

Una vez cargado simplemente hay que procesar la imagen con el método detectMultiScale que devuelve un vector de rectángulos, uno por cada coche que haya encontrado en la imagen.



Detección de coches con HAAR 1

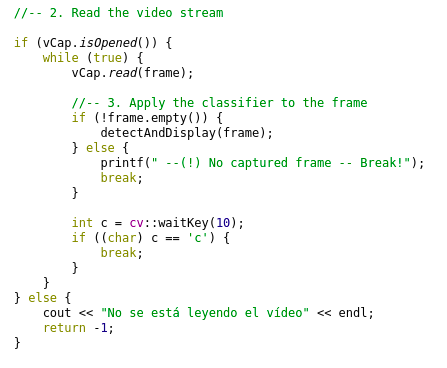
Después si se quiere se pueden dibujar los rectángulos y mostrar la imagen para comprobar que se están detectando correctamente.



Detección de coches con HAAR 2

## Parte 3: Detección de coche en secuencias de vídeo

Para esta última parte solo hay que añadir un poco de código de OPENCV que permitirá procesar un vídeo de manera que podamos analizar cada uno de los frames buscando coches. Para ello utilizaremos el objeto de OPENCV cv::VideoCapture y sus métodos (isOpened, read…), por lo demás el procesamiento de cada frame es exactamente el mismo que el que realizábamos para imágenes sueltas por lo que el código queda así:



Detección de Coches en Vídeos

# Estadísticas

# Conclusiones

Algunas de las partes de la realización de la práctica han sido bastante complicadas, en parte por el propio contenido de la asignatura que ya de por sí es a veces complejo y en parte por el lenguaje C++ , lenguaje que no habíamos utilizado hasta este año, sumado a la utilización de OPENCV. Todo esto junto hizo del comienzo de la práctica algo, por lo menos, árido.

Pero a decir verdad, una vez superada esa primera barrera que era más de desconocimiento de la herramienta que otra cosa, y en retrospectiva, consideramos que la práctica ha sido muy útil e interesante puesto que nos ha obligado a aprender c++, lenguaje que íbamos a necesitar aprender en algún momento u otro, así como el uso de distintas librerías además de permitirnos comprender mejor parte de la asignatura que luego a la hora de estudiar para el examen nos liberó de bastante carga. (Por ejemplo en toda la parte de detección de puntos de interés, votación a la Hough, descriptores, etc).