[PORTADA]

[INDICE]

# Práctica III: Detección de objetos

## Detección de coches mediante puntos de interés

### Carga de imágenes

Para realizar la carga, primero listamos los ficheros del directorio y filtramos por la extensión ‘.*jpg’*. Tras esto se cargan una a una en escala de grises con *cv2.imread* y se almacena en un objeto de tipo *Image,* que se añade a la lista de objetos Image a devolver.



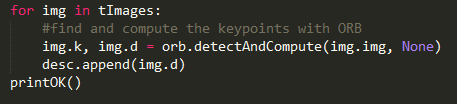
**La clase Image** nos va a permitir tener localizada en un mismo objeto toda la información relacionada con la imagen. Esto es:

* La imagen 🡪 *img*
* Los puntos de interés (que se añadirán más adelante) 🡪 *k*
* Los descriptores (que se añadirán más adelante) 🡪 *d*

### Obtención de los puntos de interés y descriptores usando la clase cv2.ORB

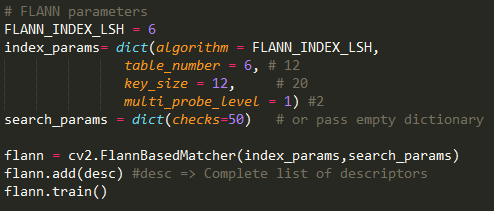
Para la obtención de los puntos de interés y descriptores, se ha utilizado un objeto ORB inicializado a cien puntos de interés, una pirámide de cuatro niveles y un factor de escala de uno.

En el siguiente bucle iteramos sobre los objetos *Image* previamente cargados y completamos para cada uno de ellos los campos *keypoints* and *descriptors* con los resultados del ORB. Aprovechamos para guardar en *desc* una lista con todos los descriptores para más adelante entrenar al clasificador.



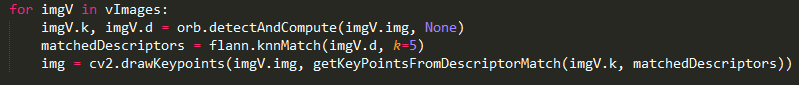
### Parametrización y entrenamiento del clasificador FLANN

Para identificar los puntos comunes en las imágenes, se ha inicializado un clasificador FLANN con un algoritmo basado en la distancia Hamming (dado que ORB devuelve descriptores binarios). Una vez añadimos los descriptores, guardados en *desc*, lo entrenamos.

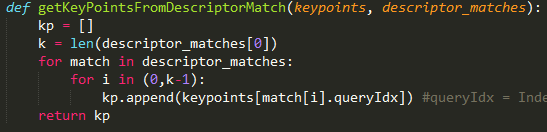


### Procesado de imágenes con el clasificador FLANN

Una vez entrenado, iteramos sobre la lista de objetos *Image* **de validación**, utilizando la función knnMatch del clasificador FLANN para obtener los cinco descriptores (*k = 5)* que más similitud guardan con las imágenes que hemos utilizado para su entrenamiento.

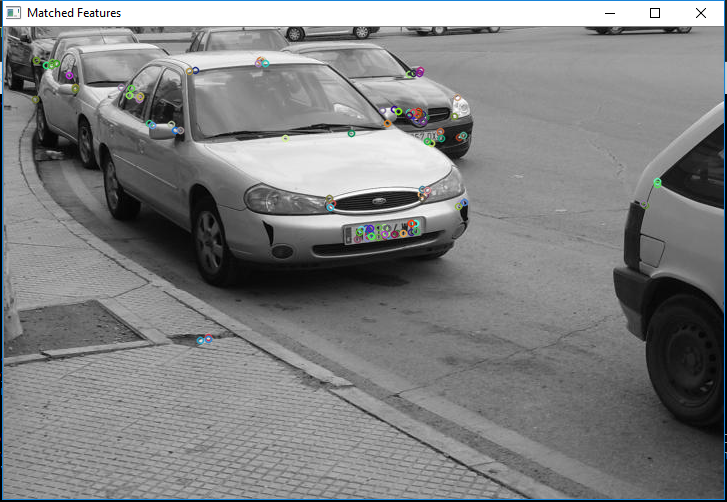


Finalmente, dibujamos los puntos de interés obtenidos sobre la imagen. Para traducir los descriptores a puntos de interés, usaremos nuestra función *getKeyPointsFromDescriptorMatch(keypoints, descriptors):*



Esta función obtiene el índice de cada descriptor accediendo al atributo *queryIdx* y devuelve una lista con todos los puntos de interés.

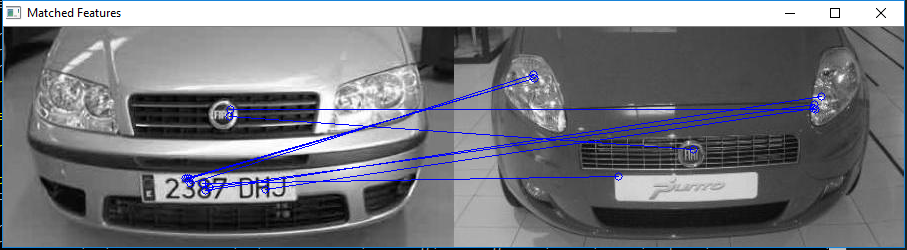
Podemos visualizar estos puntos de interés sobre la imagen para verificar su eficacia:



Lo primero que vemos es que la matrícula concentra la mayoría de puntos de interés. Esto es debido a que todos los coches comparten un mismo diseño de placa de matrícula (rectangulares y con caracteres de la misma fuente).

También hay algunos puntos que caen en la zona de incertidumbre o ruido del clasificador, como las grietas en el suelo o los reflejos en el cristal, que son erróneos o no identificativos.

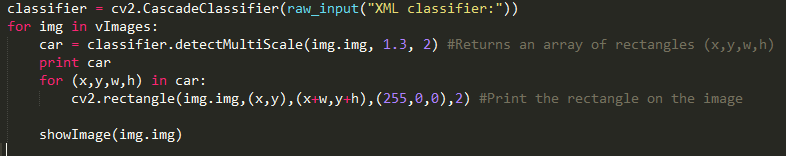
De hecho, podemos llamar a la función *drawMatches* (implementada en utils.py, dado que sólo está disponible en Python 3) para cruzar los puntos de interés similares de dos imágenes:



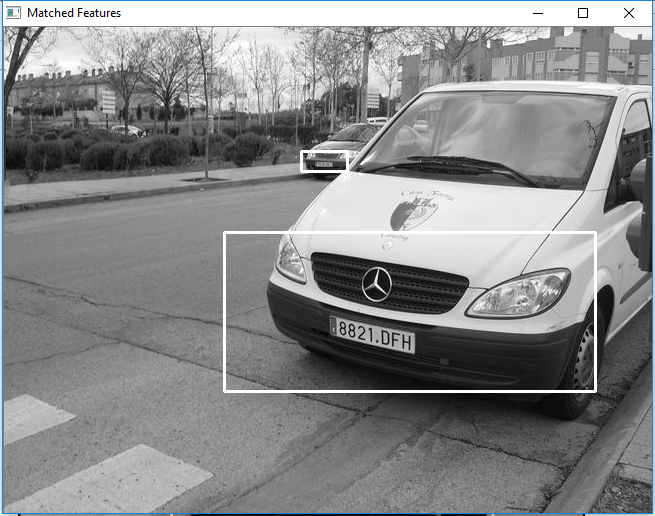
## Detección de coches usando cv2.CascadeClassifier

En esta sección vamos a utilizar los puntos de interés para delimitar con un rectángulo el área donde se encuentra la parte frontal del coche. Esta vez se nos proporciona un clasificador de tipo *CascadeClassifier* ya entrenado en el fichero “*coches.xml”*.

Primero creamos un objeto *CascadeClassifier* a partir del fichero xml proporcionado. Luego iteramos sobre el conjunto de objetos *Image* obteniendo para cada uno una lista de tuplas (*x,y,w,h*), donde cada una define un rectángulo detectado (puede haber varios coches en la imagen). Por último, dibujamos cada rectángulo en la imagen con la función *cv2.rectangle*.



Este sería un resultado con dos coches correctamente identificados:



## Detección del coche en secuencias de vídeo