VISIÓN ARTIFICIAL
Práctica 3: Detección de objetos

El principal objetivo de esta práctica es construir software de detección de objetos utilizando puntos característicos y descriptores ORB.

El software a desarrollar se gestionará a través de una aplicación desde la que el usuario podrá construir conjuntos de imágenes de los los objetos a detectar, así como visualizar el resultado de la detección en el flujo de vídeo proporcionado por la cámara. La figura 1 muestra una posible interfaz para esta aplicación.

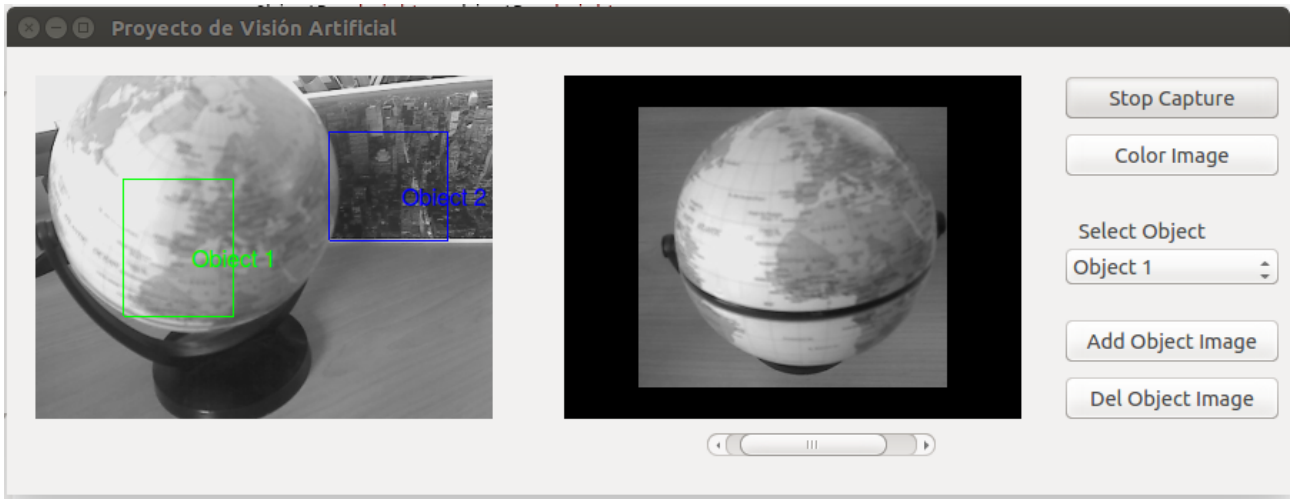

Figura 1: interfaz de la aplicación

La ventana de la izquierda muestra la imagen de la captura de cámara actual con el resultado de la detección de objetos. Cada objeto detectado se representa mediante un marco rectangular y un texto que indica el identificador del objeto detectado en esa zona de la imagen. El marco representado es el menor rectángulo ortogonal (no girado) que permite incluir todos los puntos coincidentes con el conjunto de imágenes de entrenamiento del objeto. Para que la representación sea más distintiva, cada objeto es visualizado con un color diferente.

La ventana de la derecha muestra las imágenes utilizadas para identificar cada objeto. Para visualizar las imágenes de un objeto concreto, el usuario debe seleccionar un identificador de objeto a través de la lista desplegable etiquetada como "*Select Object*" (figura 2). Una vez seleccionado, la barra de desplazamiento situada en la parte inferior le permite visualizar las distintas imágenes asociadas al objeto (figura 3).

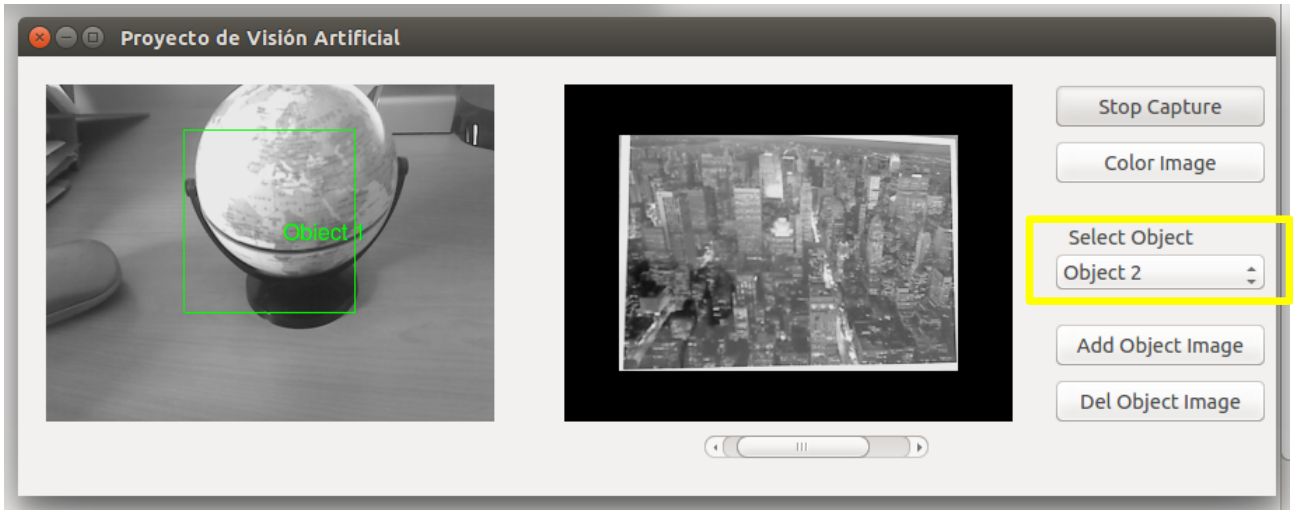

Figura 2: selección de objeto para visualización, añadir imágenes al objeto o borrar imágenes del objeto.

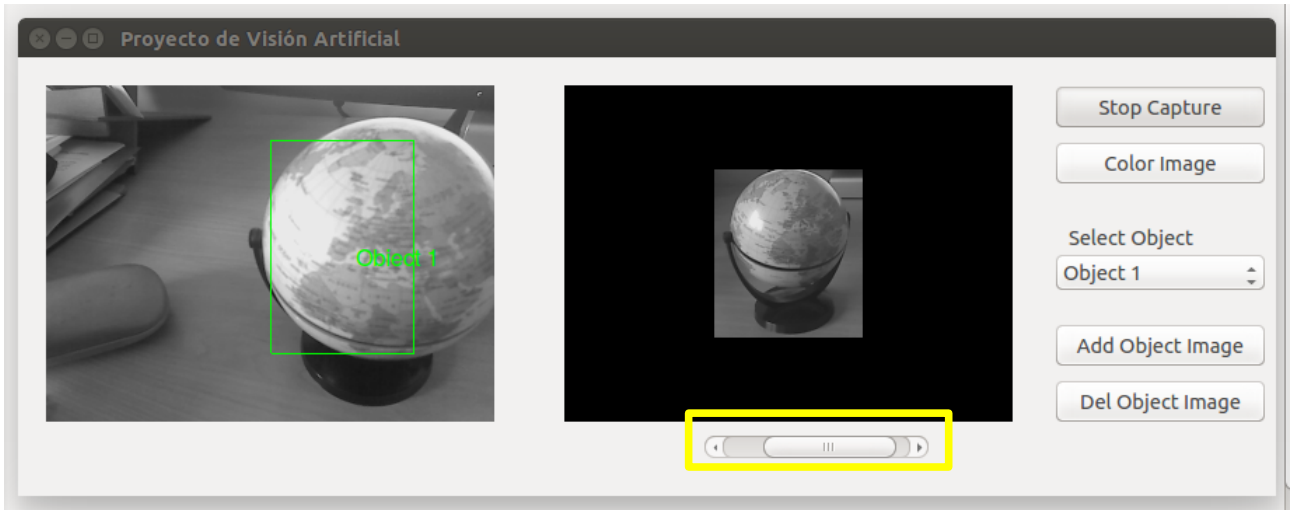

Figura 3: visualización de las imágenes asociadas a un objeto

La interfaz incluye además 2 opciones para la gestión de las listas de imágenes asociadas con los diferentes objetos a detectar. En concreto, la opción "*Add Object Image*" añade la ventana de imagen seleccionada por el usuario en la imagen de la izquierda (ver figura 4) a la lista de imágenes del objeto actual (figura 5). Asimismo, el usuario tiene la posibilidad de eliminar la imagen de objeto visible actualmente mediante la opción "*Del Object Image*".

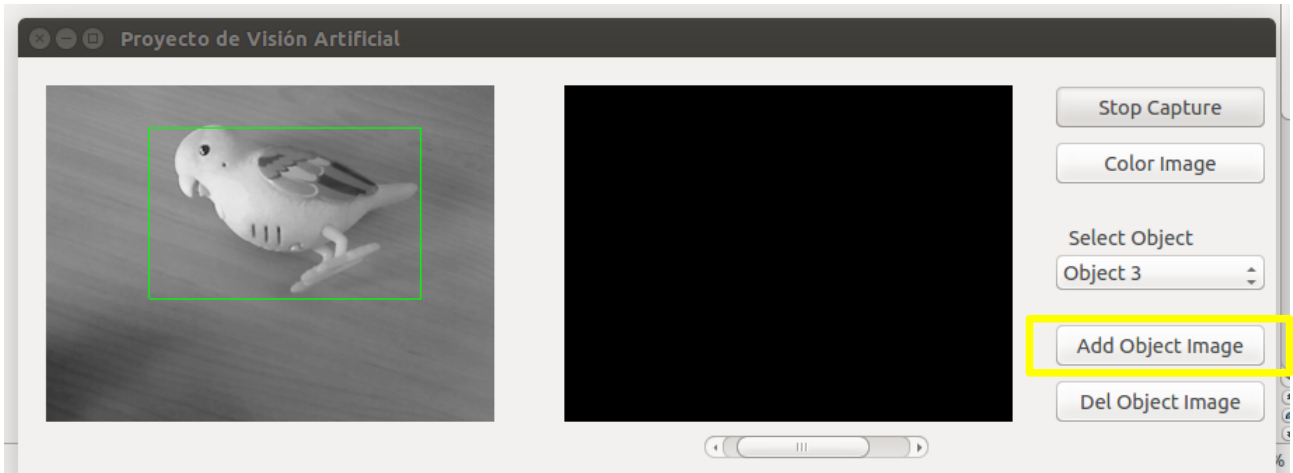

Figura 4: selección de ventana para añadir una imagen al objeto seleccionado.

Figura 5: ejecución de la opción "*Add Object Image*"

Especificación de objetivos

En su versión básica, el software de detección localiza los objetos en la imagen siguiendo dos fases:

- Obtener las correspondencias entre los puntos característicos de la imagen actual y de las imágenes de los objetos. Cada punto característico de la imagen debe asociarse al punto característico de objeto que minimiza la distancia entre descriptores. Como resultado un punto característico de la imagen se clasifica como perteneciente a un objeto determinado.
- Calcular el menor rectángulo ortogonal que permite incluir todos los puntos en correspondencia con un mismo objeto.

Para mejorar los resultados de detección, es necesario imponer un valor máximo de distancia entre descriptores. Así, sólo deben aceptarse aquellas correspondencias cuya distancia sea inferior a un valor umbral. La figura 6 muestra las correspondencias entre la imagen de la captura actual (ventana de la izquierda) y la imagen de objeto (ventana de la derecha). Se muestran en rojo los puntos característicos de la imagen cuya correspondencia con el objeto presenta un valor de distancia superior a un umbral. En verde están representados los puntos en correspondencia con el objeto cuya distancia entre descriptores es considerada aceptable. Sólo este último conjunto de puntos es tenido en cuenta para enmarcar el objeto. Como puede observarse en la figura, si no se aplicara este filtro de distancia, la zona de imagen asociada con el objeto cubriría partes que no se corresponderían con él. Asimismo, ante la no presencia del objeto, podría darse el caso de que el software de detección proporcionara una respuesta positiva en una determinada región de la imagen. Esta situación puede observarse en la figura 7.

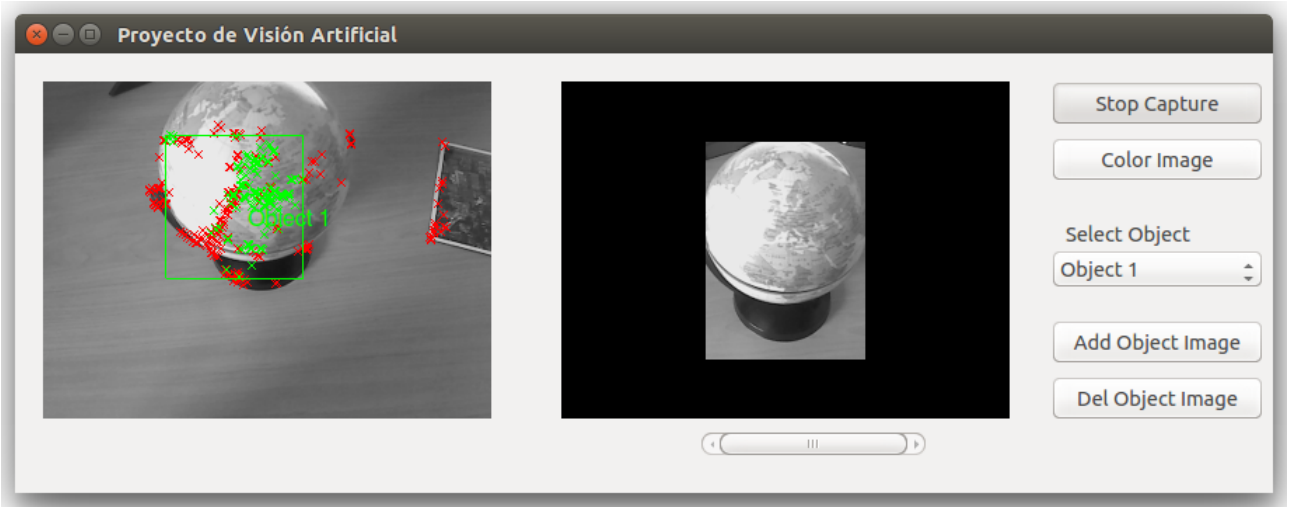

Figura 6: descartar correspondencias cuya distancia entre descriptores supera un cierto umbral.

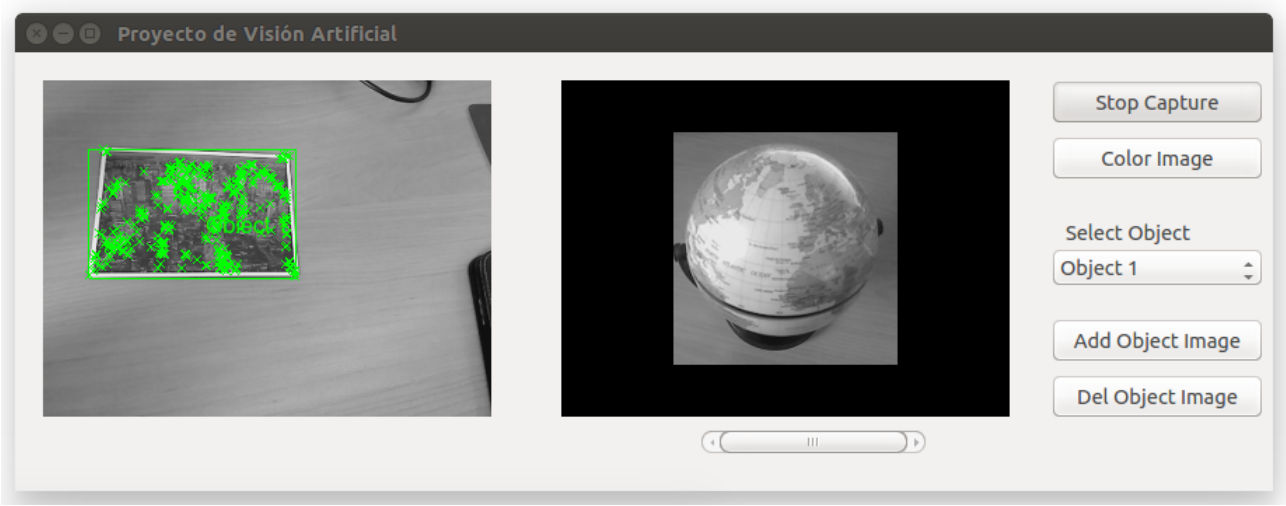

Figura 7: detección errónea del objeto provocada por la ausencia de comprobación de distancia entre descriptores.

En relación a la aplicación de gestión del software de detección, ésta debe proporcionar las diferentes opciones descritas anteriormente para permitir al usuario construir los conjuntos de imágenes que definen cada objeto, así como visualizar los resultados de detección. Se podrán definir hasta 3 objetos y cada objeto podrá tener asociado hasta 5 imágenes. Se plantea como ampliación de la aplicación la posibilidad de que los conjuntos de imágenes de objeto puedan ser guardados en fichero antes cerrar la aplicación, así como la inclusión de una opción que permita cargar conjuntos de imágenes de objeto desde fichero para su posterior detección.

Ampliaciones del software de detección

Se proponen dos ampliaciones relacionadas con la mejora en la detección de los objetos. Éstas son las siguientes:

- *Comprobación de dispersión de puntos característicos pertenecientes a un mismo objeto*: los puntos característicos pertenecientes a un mismo objeto deben estar concentrados en una misma región de la imagen. Así, aquellos puntos cuya distancia a un punto de referencia se considere excesiva, deberán descartarse (ver figura 8). Para realizar esta comprobación, se escogerá como punto de referencia aquel que tomado como centro de una circunferencia

permita concentrar con el menor radio la mitad de los puntos clasificados como pertenecientes al objeto. Los restantes puntos se iran añadiendo en orden de distancia al punto de referencia hasta que se dé la situación en la que la inclusión de un nuevo punto implique aumentar excesivamente el área de la región asociada con el objeto (por ejemplo, al doble de su tamaño actual). En tal caso, el punto en cuestión, así como los no analizados hasta ese momento, se descartarán.

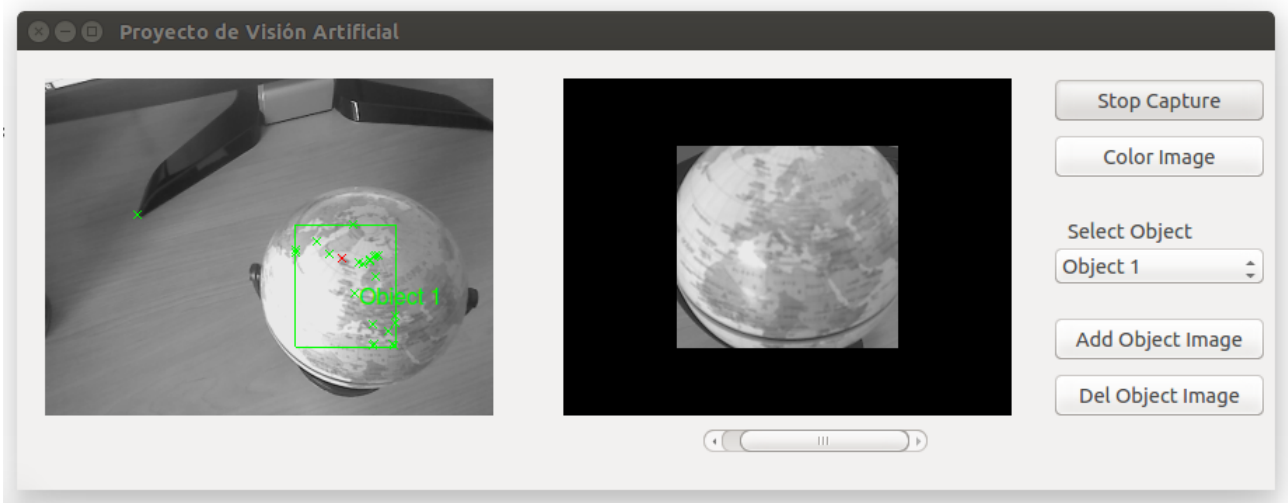

Figura 8: descartar correspondencias dispersas.

- *Comprobación de solapamiento de objetos*: si dos regiones de objeto se encuentran altamente solapadas, es bastante probable que una de ellas no se corresponda con el objeto detectado. Para realizar esta comprobación se medirá el solapamiento como la relación entre el área común de dos regiones y el área de la menor región. Si el solapamiento es superior a un cierto porcentaje, se descartará la región que concentre el menor número de puntos característicos (ver figura 9).

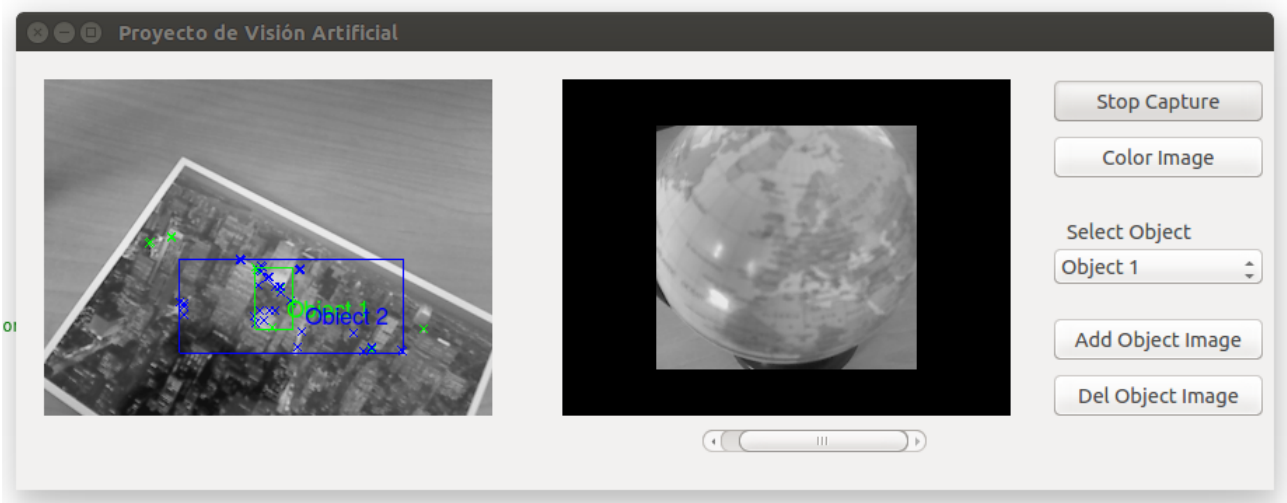

Figura 9: ante regiones de objeto altamente solapadas, seleccionar aquella que concentra el mayor número de correspondencias.

Utilidades

OpenCV proporciona una serie de clases para la detección de puntos característicos, así como para el cálculo de descriptores y la búsqueda de correspondencias entre conjuntos de imágenes. En relación a la detección de puntos característicos y a la obtención de los correspondientes descriptores ORB, el módulo *features2D* incluye las siguientes utilidades:

Clase ORB

- Clase heredada de *Feature2D*.
- Creación de objeto ORB:
  - ***static Ptr<ORB> cv::ORB::create****(int* ***nfeatures****=500, float* ***scaleFactor****=1.2f, int* ***nlevels****=8, int* ***edgeThreshold****=31, int* ***firstLevel****=0, int* ***WTA_K****=2, int* ***scoreType****=ORB::HARRIS_SCORE, int* ***patchSize****=31, int* ***fastThreshold****=20)*
    - ***nfeatures***: número máximo de características a mantener.
    - ***scaleFactor*:** ratio de diezmado para construir la pirámide.
    - ***nlevels***: número de niveles de la pirámide
    - ***edgeThreshold***: tamaño del borde a partir del cual no se detectan características.
    - ***firstLevel***: primer nivel sobre el que aplicar la detección.
    - ***WTA_K***: número de puntos que intervienen en cada test para extraer el descriptor *rBRIEF*.
    - ***scoreType***: tipo de método utilizado para "puntuar" las características.
    - ***patchSize***: tamaño de la región utilizada para extraer el descriptor de cada característica.
    - ***fastThreshold***: umbral del método *FAST* para detección de características.
- Métodos principales:
  - *void* ***detect****(const Mat&* ***image****, vector<KeyPoint>&* ***keypoints****, const Mat&* ***mask****=Mat())*: detecta los puntos característicos de la imagen *image* y los devuelve en *keypoints*.
  - *void* ***compute****(const Mat&* ***image****, vector<KeyPoint>&* ***keypoints****, Mat&* ***descriptors****)*: genera los descriptores de las regiones de *image* indicadas en *keypoints* y los devuelve en *descriptors*. Cada descriptor se almacena en una fila de un objeto de tipo *Mat*. El índice de fila de un descriptor coincide con el índice dentro de *keypoints* del punto característico asociado.

Clase KeyPoint

- Estructura de datos para almacenar puntos característicos.
- Atributos:
  - ***Point2f pt***: coordenadas del centro de la región en la imagen (pt.x, pt.y).
  - ***float size***: diámetro de la región.
  - ***float angle***: orientación principal de la región.
  - ***float response***: respuesta de la región durante la detección.
  - ***int octave***: capa de la pirámide en la que se ha detectado
  - ***int class_id***: identificador utilizado para agrupar características pertenecientes a un mismo objeto.

En relación al cálculo de correspondencias entre descriptores, el módulo *features2D* incluye, entre otras, las siguientes utilidades.

Clase BFMatcher

- Clase heredada de DescriptorMatcher.
- Constructor:
  - ***BFMatcher****(int* ***normType****=NORM_L2, bool* ***crossCheck****=false)*
    - ***normType***: tipo de distancia utilizada para evaluar las correspondencias entre características: *NORM_L1, NORM_L2, NORM_HAMMING, NORM_HAMMING2.*
    - ***crossCheck***: busca correspondencias consistentes en ambos sentidos.
- Métodos principales:
  - ***add****(const vector<Mat>& descriptors)*: añade nuevos grupos de descriptores a una colección de descriptores de entrenamiento. Cada elemento de la lista se corresponde con un conjunto de descriptores pertenecientes a la misma imagen de entrenamiento. Si se utiliza una colección a través de este método, no hay que indicar el parámetro *trainDescriptors* en los métodos de *matching* que se especifican a continuación.
  - ***clear()***: borra la colección de descriptores de entrenamiento.
  - *void* ***match****(const Mat&* ***queryDescriptors[****, const Mat&* ***trainDescriptors]****, vector<DMatch>&* ***matches****, const Mat&* ***mask****=Mat())*: calcula las correspondencias entre *queryDescriptors* y *trainDescriptors* y devuelve el resultado en *matches*. Si se ha especificado una colección de descriptores de entrenamiento, la búsqueda de correspondencias se realiza entre *queryDescriptors* y los descriptores de la colección.
  - *void* ***knnMatch****(const Mat&* ***queryDescriptors[****, const Mat&* ***trainDescriptors]****, vector<vector<DMatch>>&* ***matches****, int* ***k****, const Mat&* ***mask****=Mat(), bool* ***compactResult****=false)*: por cada descriptor de *queryDescriptors*, encuentra los *k* descriptores más similares en *trainDescriptors*.
  - *void radiusMatch(const Mat&* ***queryDescriptors[****, const Mat&* ***trainDescriptors]****, vector<vector<DMatch>>&* ***matches****, float* ***maxDistance****, const Mat&* ***mask****=Mat(), bool* ***compactResult****=false)*: proporciona las correspondencias entre pares de descriptores que no superan una distancia máxima.

Estructura DMatch

- Almacena la información de correspondencia entre dos descriptores.
- Atributos principales:
  - ***queryIdx***: índice del descriptor de la lista de descriptores de la imagen de búsqueda (parámetro *queryDescriptors*).
  - ***trainIdx***: índice del descriptor de la lista de descriptores de la imagen de entrenamiento (parámetro *trainDescriptors*) o de una imagen de la colección de entrenamiento especificada con el método *add*.
  - ***imageIdx***: índice de la imagen de entrenamiento en la que se ha encontrado la correspondencia (índice dentro de la colección especificada en el método *add*).
  - ***distance***: distancia entre los descriptores del par.
- Para averiguar cuáles son los puntos característicos asociados con una correspondencia a partir de esta estructura se utilizan los atributos de la siguiente forma:
  - *queryIdx* coincide con el índice del punto característico de la imagen dentro del vector *keypoints* especificado en el método *compute* de ORB. Este índice permite directamente

indexar dicho vector para obtener las propiedades de un punto característico de la imagen en correspondencia con un punto característico del objeto.

- Si no existe una colección, el atributo *trainIdx* es el índice dentro del vector *keypoints* asociado con la imagen de objeto.
- Si se ha especificado una colección, es necesario combinar el atributo *trainIdx* junto con *imageIdx* para acceder al punto característico del objeto de una correspondencia. En concreto, *imageIdx* permite identificar la imagen de objeto asociada con la correspondencia. Así, el índice almacenado en este atributo coincide con el índice de la imagen de entrenamiento dentro del vector especificado en el método *add*. Esto permite conocer cuál es la imagen de objeto asociada con la correspondencia. Para conocer el punto caracteristico concreto dentro de la imagen de objeto, se utiliza el atributo *trainIdx* como índice del vector *keypoints* de dicha imagen.