VISIÓN ARTIFICIAL
Práctica 3: Detección de objetos

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

El software a desarrollar se gestionará a través de una aplicación desde la que el usuario podrá seleccionar 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 actual de cámara 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 se obtiene de aplicar una transformación a las esquinas que delimitan el objeto para obtener las nuevas posiciones de dichas esquinas en la imagen. Para que la representación sea más distintiva, cada objeto es visualizado con un color diferente.

La ventana de la derecha muestra la imagen utilizada para identificar cada objeto. Para visualizar la imagen de un objeto concreto, el usuario debe seleccionar un identificador de objeto a través de la lista desplegable etiquetada como "*Select Object*".

Figura 2: selección de ventana para añadir una imagen al objeto seleccionado.

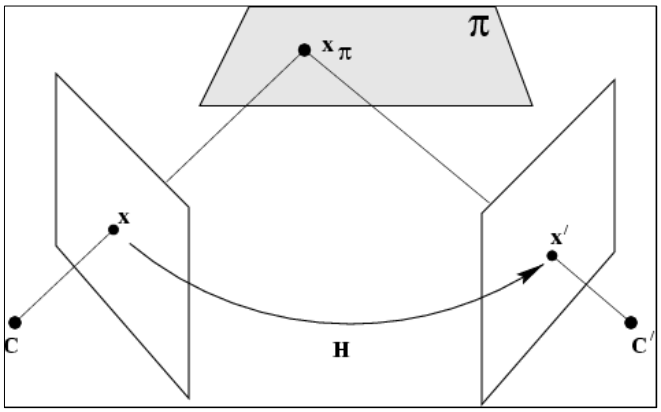
La interfaz incluye además 2 opciones para la gestión de las imágenes asociadas con los diferentes objetos a detectar. En concreto, la opción "*Add Object Image*" asocia la ventana de imagen seleccionada por el usuario en la imagen de la izquierda (figura 2) al objeto actual (figura 3). Si ya existe una imagen para ese objeto, se reemplazará la imagen anterior por la nueva. Asimismo, el usuario tiene la posibilidad de eliminar la imagen asociada con un objeto mediante la opción "*Del Object Image*". Cuando se utilice esta segunda opción, el objeto asociado ya no podrá ser detectado.

Figura 3: 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 3 fases:

- Obtener las correspondencias entre los puntos característicos de la imagen actual y los de las imágenes de los objetos.
  Para mejorar la invarianza de la detección antes cambios de escala, por cada imagen de objeto se crearán 3 escalas diferentes. Para ello, se tomará la imagen seleccionada por el usuario como escala intermedia y se creará, a partir de ella, una imagen de menor escala (p.e. 75% del tamaño original) y otra de mayor escala. Durante el cálculo de correspondencias, cada punto de imagen podrá ponerse en correspondencia con hasta 3 imágenes de objeto. Se seleccionarán sólo aquellas correspondencias cuya distancia sea inferior a un umbral para evitar falsas asociaciones de puntos.
- A partir de las correspondencias anteriores, seleccionar para cada objeto la escala con mayor número de correspondencias. La imagen de objeto y las correspondencias que se utilizarán en la siguiente fase serán las de la escala seleccionada.
- Por cada objeto:
  - Calcular la transformación que relaciona los puntos del objeto con los puntos homólogos en la imagen actual. Para objetos planos esta transformación viene dada por una matriz de 3x3 denominada homografía.
    Dadas dos vistas de un plano, para cualquier punto de dicho plano x_Π, se verifica $x' = Hx$, siendo x y x' proyecciones (posiciones de píxel) de x_Π en las dos vistas, expresadas en coordenadas homogéneas:



- Aplicar la homografía obtenida anteriormente a las esquinas del objeto para obtener la posición de estas esquinas en la imagen actual. Las 4 posiciones obtenidas definen el marco del objeto en la imagen.

En relación a la aplicación de gestión del software de detección, ésta debe proporcionar las diferentes opciones descritas anteriormente para permitir al usuario generar los conjuntos de imágenes que definen a los objetos, así como visualizar los resultados de detección. Se podrán definir hasta 3 objetos. Se plantea como ampliación de la aplicación la posibilidad de que las imágenes de objeto puedan ser guardadas en fichero antes de cerrar la aplicación, así como la inclusión de una opción que permita cargar imágenes de objeto desde fichero para su posterior detección. También se plantea como ampliación aumentar el número de objetos a detectar, manteniendo resultados aceptables de reconocimiento.

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.
  - *void* ***detectAndCompute****(const Mat&* ***image****, const Mat&* ***mask****, vector<KeyPoint>&* ***keypoints****, Mat&* ***descriptors****)*: detecta los puntos característicos y calcula sus descriptores.

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:
  - ***static Ptr<BFMatcher> cv::BFMatcher::create****(int* ***normType****=NORM_L2, bool* ***crossCheck****=false)*: creación de objeto BFMatcher para el cálculo de correspondencias (sustituye al constructor).
  - ***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* o en una colección creada previamente.
- *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.

Clase 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*.
  - ***imgIdx***: 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 *imgIdx* para acceder al punto característico del objeto de una correspondencia. En concreto, *imgIdx* 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 característico concreto dentro de la imagen de objeto, se utiliza el atributo *trainIdx* como índice del vector *keypoints* de dicha imagen.

Desarrollo de una solución

Creación de una colección (su aplicación será necesaria cada vez que se modifique la lista de objetos)

- Borrar la colección de *BFMatcher*.
- A partir de la lista de objetos, por cada objeto:
  - Generar las distintas escalas a partir de la imagen original del objeto.
  - Obtener los puntos característicos (*keypoints*) de cada escala y guardarlos en un vector.
  - Obtener los descriptores de cada escala y guardalos en un vector.
  - Añadir el vector de descriptores a la colección actual de *BFMatcher*.

Procesamiento por cada nueva captura (sólo debe llevarse a cabo si se han añadido objetos)

- Obtener el vector de puntos característicos de la imagen actual.
- Obtener su conjunto de descriptores.
- Poner en correspondencia los descriptores anteriores con la colección, obteniendo por cada punto característico de la imagen tantos candidatos como escalas de objeto (método *knnMatch* de *BFMatcher*)
- Descartar las correspondencias cuya distancia entre descriptores sea superior a un umbral. A partir de las correspondencias aceptadas, crear por cada objeto y escala una lista de correspondencias. El índice de objeto y escala de cada correspondencia se obtendrá del indice de imagen de la colección (*imgIdx*).
- Por cada objeto:
  - Seleccionar la escala con mayor número de correspondencias.
  - Para la escala seleccionada:
    - Separar las parejas en correspondencias en dos listas: lista de puntos de imagen y lista de puntos de objeto (dos puntos emparejados deben tener el mismo índice en las dos listas).
    - A partir de las listas anteriores, calcular la homografía que transforma los puntos del objeto en puntos de imagen (función *findHomography* de OpenCV). Sólo podrá aplicarse si existe un número mínimo de puntos en las dos listas (p.e. 10 puntos).
    - Si se ha obtenido un homografía válida, aplicarla a las cuatro esquinas de la imagen del objeto en la escala seleccionada para obtener las cuatro esquinas del objeto en la imagen actual (función *perspectiveTransform* de OpenCV). Dibujar en pantalla, del color identificativo del objeto, el rectángulo definido por los 4 puntos anteriores y mostrar el texto identificativo del objeto en su posición central.

Ampliaciones

Además de las ampliaciones que se sugieren en la sección de "*Especificación de objetivos*", se propone realizar ampliaciones destinadas a mejorar la detección. En este sentido, una posible mejora consistiría en descartar aquellos puntos de imagen que se hayan puesto en correspondencia con más de un objeto. Así, si se dan varias correspondencias (con valor de distancia entre descriptores aceptable) en las que un mismo punto de imagen se encuentra emparejado con diferentes objetos, estas correspondencias no serán consideradas válidas y, por lo tanto, no se tendrán en cuenta en fases posteriores del procesamiento.