TP détection de points clés

0 - Préliminaires :

Téléchargez les deux jeux de données set1.zip et set2.zip.
Nous allons faire un programme qui prendra en arguments une vidéo et une liste d'images.
Dans un premier temps, faites en sorte que le programme affiche la première image dans une fenêtre "object" et lise la vidéo dans une fenêtre "scene".

1 - ORB :

Nous allons utiliser la classe cv::ORB, qui hérite à la fois de FeatureDetector et de DescriptorExtractor.
Cette classe permet donc à la fois de détecter les points d'intérêt dans l'image, via la méthode detect , puis dans un second temps, de générer les descriptions des points d'intérêt, via la méthode compute.

Modifiez votre programme pour qu'il affiche les points détectés sur l'image, ainsi que sur la vidéo.

2 - Mise en correspondance :

Maintenant que vous avez des points d'intérêts, ainsi que leur description, il est temps de les mettre en correspondance. Pour cela vous pouvez utiliser la classe cv::BFMatcher, qui va simplement, pour chaque point d'intérêt de la scène, calculer la distance de son descripteur avec tous ceux de l'objet, pour trouver le plus proche.

Il faut faire attention ici a quelle mesure utiliser pour calculer la distance entre deux descripteurs. Par défaut, celle utilisée est la norme euclidienne, qui est adaptée aux vecteurs de réels (comme ceux décrivant les SIFT ou SURF). Pour les descripteurs binaires, il vaut mieux utiliser la distance de Hamming.

Modifiez votre programme pour qu'il affiche les correspondances entre l'objet et la scène.

3 - Détourage :

Un fois les appariements disponibles, il est possible de calculer la transformation optimale pour passer de l'objet à la scène.
Pour cela, utilisez d'abord la fonction findHomography, puis, au moyen de la matrice de transformation obtenue, projetez les 4 coins de l'objet dans la scène, au moyen de la fonction perspectiveTransform, et reliez les par des lignes.

4 - Multi Matching :

On va maintenant gérer plusieurs images a détecter. Utiliser les méthodes add et train de BFMatcher, pour enregistrer successivement les descripteurs des différentes images.
Chaque point d'intérêt de la scène sera donc mis en correspondance avec un unique point, et donc une seule image parmis l'ensemble passé en paramètres. Certaines images auront donc beaucoup plus d'appariements que d'autres.
Au dessus d'un certain nombre d'appariements avec une image, cela veut dire que celle-ci est présente dans la scène, vous devrez alors la détourer, par le même procédé que dans l'étape précédente, en utilisant une couleur distincte pour chaque 'image.

5 - Amélioration des résultats :

Libre à vous d'améliorer les résultats. Plusieurs voies sont possibles :

- améliorer la détection des points, en jouant sur les paramètres de ORB
- filtrer les matchings pour ne garder que les meilleurs (ceux, tels que le second meilleur choix est loin derrière.) Vous pouvez vous amuser avec knnMatch et radiusMatch, par exemple.
- essayer d'obtenir une meilleure projection, en jouant sur les paramètres de findHomography, ou en ne lui fournissant que des points cohérents.
- détecter que la projection en elle même est cohérente (quasiment un rectangle).

Exemples de résultats acceptables, bien que largement perfectibles :
set 1 : https://youtu.be/fk4V5Ej_sCU
set 2 : https://youtu.be/yZ0Qpj0p9QE
Pour l'affichage, des fonctions existent :
http://docs.opencv.org/2.4/modules/features2d/doc/drawing_function_of_keypoints_and_matches.html