Année 2021/2022
Master Informatique Visuelle

Série de travaux pratiques n°2
Vision par Ordinateur

Tools for OpenCV-Python programming

Finding the SIFT keypoints and descriptors

```
sift = cv2.SIFT_create()
kp, des = sift.detectAndCompute(image, None)
```

cv2.SIFT.detectAndCompute(): Detects keypoints and computes their descriptors

Input: Image, input 8-bit grayscale image.
Output:
Keypoints: The detected keypoints. 1-by-N structure array with the following fields:

- pt coordinates of the keypoint [x,y]
- size diameter of the meaningful keypoint neighbourhood
- angle computed orientation of the keypoint (-1 if not applicable); it's in [0,360) degrees and measured relative to image coordinate system (y-axis is directed downward), i.e in clockwise.
- Response: the response by which the most strong keypoints have been selected. Can be used for further sorting or subsampling.
- octave octave (pyramid layer) from which the keypoint has been extracted.
- class_id object class (if the keypoints need to be clustered by an object they belong to).

Descriptors: Computed descriptors. Output concatenated vectors of descriptors. Each descriptor is a 128-element vector,

Feature Matching

To match SIFT features in one image with others, we may use the **Brute-Force matcher** in OpenCV

Brute-Force Matcher
Takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. The closest one is returned.

cv2.BFMatcher(parameter_1, parameter_2) → create the BFMatcher object

parameter_1: normType. It specifies the distance measurement to be used. By default, it is cv2.NORM_L2. It is good for SIFT, SURF etc (cv2.NORM_L1 is also there).

Parameter_2: crossCheck. Is false by default
If it is true, Matcher **returns only those matches** with value (i,j) such that i-th descriptor in set A has j-th descriptor in set B as the best match and vice-versa. That is, the two features in both sets should match each other. It provides consistant result, and is a good alternative to ratio test proposed by D.Lowe in SIFT paper.

BFMatcher.match() is a method that returns the best match.
BFMatcher.knnMatch() is a method that returns *k* best matches where k is specified by the user. It may be useful when we need to do additional work on that.

cv2.drawMatches() draws the matches. It stacks two images horizontally and draws lines from first image to second image showing best matches.
cv2.drawMatchesKnn which draws all the k best matches. If k=2, it will draw two match-lines for each keypoint. So we have to pass a mask if we want to selectively draw it.
cv2.BFMatcher.knnMatch():
returns K best matches, where k is specified by the user.
Matches=cv2. **BFMatcher.knnMatch** (query, trains, k[, mask[, compactResult]])

Matches: the matches. Each matches[i] is k or less matches for the same query descriptor
query: Query set of descriptors.
trains: Train set of descriptors. This set is not added to the train descriptors collection stored in the class object.
k: Count of best matches found per each query descriptor or less if a query descriptor has less than k possible matches in total.
Mask: Mask specifying permissible matches between an input query and train matrices of descriptors.
compactResult: Parameter used when the mask (or masks) is not empty. If false, the matches vector has the same size as queryDescriptors rows. If compactResult is true, the matches vector does not contain matches for fully masked-out query descriptors.

Exercice 1:

1- En utilisant les outils présentés précédemment, écrire le programme en Python qui lit deux images (requête, modèle) et calcule les descripteurs SIFT. Visualisez les attributs du point et son descripteur. A noter que l'image requête (query) est un objet de l'image modèle (train).
2- Il s'agit ensuite de rechercher l'image requête dans l'image modèle en utilisant la mise en correspondance des descripteurs SIFT.

Les figures 1 et 2 illustrent ce qui est demandé.

Figure 1. Images requête et modèle (query and train)

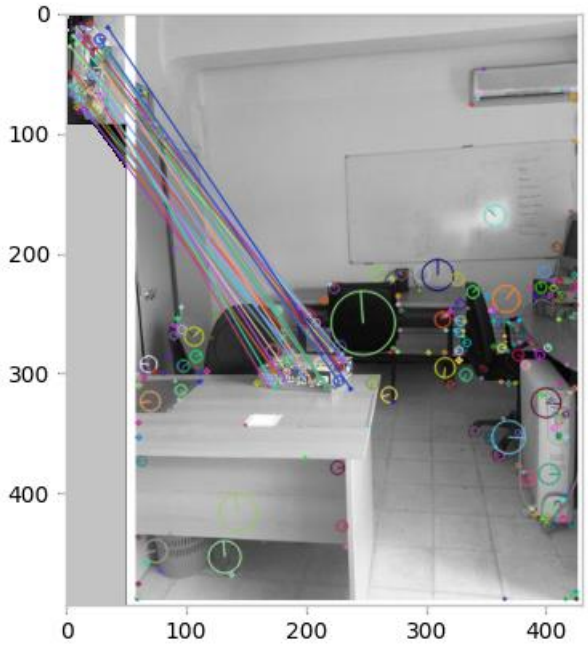

Figure 2. Output recherché

Exercice 2:

Nous disposons d'une base d'images de véhicules autorisés à entrer au parking d'un établissement. Un traitement automatique ou manuel nous permet d'isoler les plaques d'immatriculation. Les données récupérées constituent la dataset « train ».
Nous voulons vérifier si le véhicule qui se présente en entrée du parking est autorisé ou non. Pour cela nous ferons une identification de son matricule.
Une des solutions à étudier dans ce projet et d'étudier si le descripteur SIFT à lui seul permet de réaliser cette tâche. Il s'agit de vérifier si l'image requête (test) contient une région identique à une région (matricule) modèle.

Travail demandé :

1- Pour les images fournies (query, train), fournir le code qui recherche pour chaque image requête l'image modèle la contenant.
2- Proposer une framework pour :
- L'acquisition des données
- Constitution de la base modèle
- Traitement de l'image test.
- 3 - Application