Reconnaissance d'objets avec « SURF »
Object recognition using « SURF »

1 Objectif / Purpose

L'objectif est de reconnaître des cartes, en utilisant des cartes de références (dix, valet, dame et roi de carreau). Si dessous, la carte de référence est le « roi de carreau » : dans chacune des autres images, on souhaite détecter s'il y a ou non un roi de carreau, et également le localiser dans l'image.

English : The purpose is to recognize cards (10, Jack, Queen or King) using reference cards. In hereabove Figures, the reference card is the « King » of diamonds : in each other images, one wants to detect if therefore such a « King », and also compute its location (center of mass).

			
Image de référence : « roi de carreau »	1) Image contenant un « roi de carreau ».	2) Image contenant un « roi de carreau », mais aussi d'autres cartes	3) Image ne contenant pas de « roi de carreau ».

2 Approche considérée / Considered approach

Ces cartes ont toute la même forme (rectangulaire), on ne peut donc pas les distinguer (et donc les identifier) sur critère morphologique. De plus, la segmentation des cartes ne semble pas évidente. L'idée va consister à utiliser des descripteurs spécifiques « locaux » ne nécessitant pas la segmentation préalable, mais permettant de caractériser puis identifier les cartes par rapport à la « richesse » des détails (détail des dessins représentant notamment le valet, la dame et le roi).

Nous allons utiliser les descripteurs associés à l'algorithme SURF (« speed up robust features »): cet algorithme calcule des « keypoint », permet de caractériser numériquement certains points « marquants » d'une image. Par marquant, on entend des pixels associés à des détails d'une image. Un « keypoint » est caractérisé par des coordonnées (un point particulier de l'image) et est associé à un descripteur (vecteur de valeurs numériques caractérisant le voisinage de point, e.g. les intensités, les variations d'intensité, les textures,...). L'algorithme choisi pour détecter les « keypoints » et vecteurs numériques associés est le « SURF » , couramment utilisé en vision par ordinateur, et pas seulement pour du recalage et de la réalité augmentée (e.g. reconnaissance, suivi,...).

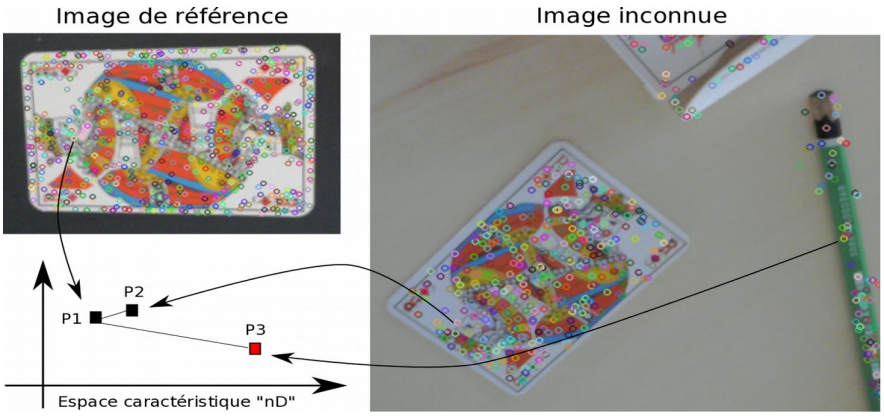

Figure 1: Keypoints

La figure 1 donne les keypoints calculés sur l'image de référence et l'image inconnue. Si l'image inconnue a beaucoup de « keypoints » similaires aux « keypoints » de l'image de référence, on pourra conclure que l'image inconnue contient l'objet présent dans l'image de référence. L'idée est donc de comparer tous les keypoints de

l'image inconnue à tous les keypoints de l'image de référence. Attention, on ne compare pas les coordonnées en « pixels » des keypoints, mais les caractéristiques. Ceci est illustré par la figure 1 dans le cas d'un vecteur caractéristique de taille 2 (espace caractéristique de dimension 2), mais en pratique, un keypoint est de beaucoup plus grande dimension. Dans le cas de l'image inconnue, le premier « keypoint » détecté a pour coordonnées [105,200] et pour un vecteur caractéristique de taille 64 : [-0.00010868 -0.00073558 0.00012374 …….. 0.00077437].

Dans la figure 1, on voit que les keypoints associés à P2 est similaire à P1 (proche) alors que P3 n'est pas similaire à P1 (éloigné), il semble logique de conclure que P2 est point du « roi de carreau » et non P3. La figure 2 affiche les correspondances détectées avec les « keypoints » de l'image de référence, et ceci pour 2 images inconnues. On constate qu'en présence d'un « roi de carreau » dans l'image inconnue, le nombre de correspondance est largement plus élevé (même si certaines correspondances sont fausses).

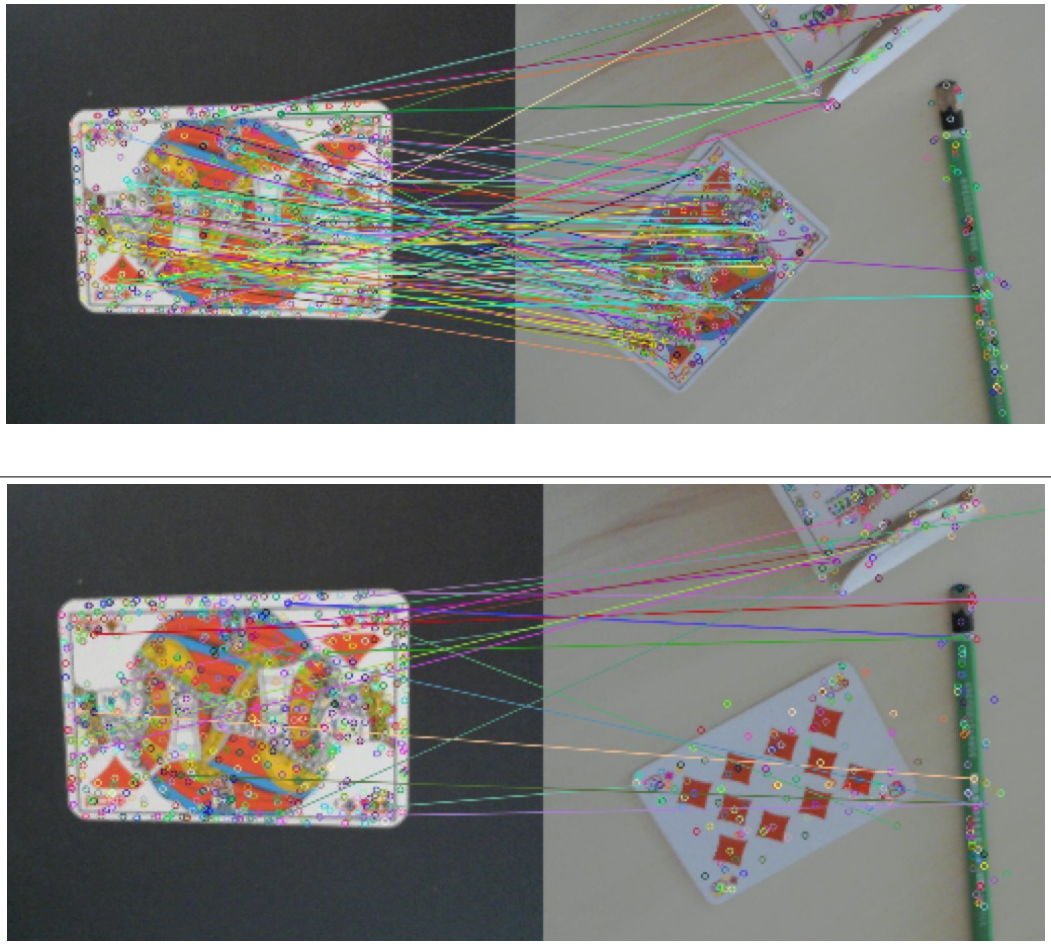

Figure 2 : les « keypoints » similaires sont reliés pour 2 images inconnues. Il y a davantage de correspondances en haut (présence d'un « roi de carreau » similaire au modèle de référence).

Tester le caneva fourni, qui illustre comment calculer les « keypoints » avec opencv, et comment comparer les « keypoints » entre deux images et détecter le nombre de correspondances selon un critère donné.

English : Recognition will be based on local descriptors, without requiring any preliminary segmentation (as done shape-based recognition) : local descriptors will embed information regarding local details (local variation of grey-levels). Descriptors will be those generated by the well-known SURF algorithm (« speed up robust features »). SURF computes « keypoints » : a « keypoint » is associated to a coordinate (particular pixel of interest in the image) and also to descriptor (vector of numerical values embedding local information mainly related to intensities and variations of intensity). Figure 1 shows « keypoints » computed on the reference image and on the « unknown » image. If the unknown image has a lot of « keypoints » whose descriptors are similar to descriptors of the reference image, one concluded the reference object is present is the unknown image. Warning : keypoints are compared using the feature (description) vector only, and not the location of the keypoint (i.e. not the coordinates of the underlying pixel). This is illustrated by Figure 1, assuming 2D feature vectors : in practice,

feature vectors depict larger dimensions (e.g. vector length of 64 : [-0.00010868 -0.00073558 0.00012374 …….. 0.00077437].).

In Figure 1, the feature vector associated to keypoint P2 is similar to the one related to P1 (there are close in the 2D feature space), while it is not the case for P3 : this appears coherent with the fact that P2 corresponds to a pixel of the « King » card (as for P1), while P3 is associated to another object. Figure 2 show matching keypoints (i.e. descriptors are close) : both « king » cards depict mainly similarities → one cand conclude that is « king » card is present in unknown image (left image).

Execute the provided python script, illustrating how keypoints can be computed using opencv, how they can be compared and how the number of match can be estimated.

3 Exercice 1 : reconnaissance / First exercise : recognition

On dispose d'images de référence dans le répertoire « DataLearning » (roi, dame, valet, dix). Pour chaque image de référence (prenons l'exemple du « roi »), on va parcourir la base de test (répertoire « DataTest ») : si l'image de test contient un roi, on va sauvegarder cette image dans le répertoire [roi], avec le nom [roi/roi_i.png] (« i » est un indice qui augmente à chaque nouvelle image de test). Sinon, on va quand même la sauvegarder, mais sous le nom [roi/not_roi_i.png]. Ceci permettra de disposer de 4 répertoire [roi/], [dame/],… , chacun contenant toutes les images de la base de test : on pourra alors aisément vérifier si la reconnaissance c'est bien passée.

Pour décider si deux images correspondent, on se basera sur le nombre de « bonnes correspondances » : s'il y a assez de « bonnes correspondances », on considérera que l'image inconnu contient un objet de type de l'image de référence (e.g. « roi de carreau »). Il y a donc un « seuil de bonnes correspondances » à définir : à vous de trouver, empiriquement, le seuil approprié. Vous pourrez aussi définir un seuil spécifique à chaque image de référence. Pour déterminer ce(s) seuil(s), pensez à d'abord afficher le nombre de « correspondances » pour chaque comparaison : ceci vous aidera à intuitivement définir un seuil « raisonnable »

English : One has reference images in the « DataLearning » directory (10, jack, queen and king). For each reference image (let us consider the example of the « king »), one will run over the set of test images (« DataTest » directory) : if a test image contains a king, one will save this image in the « King » directory, with the filename named « king_i.png » (« i » to be increased with the number of discovered kings). Otherwise, one will save in « King » directory but with the filename named « not_king_i.png ». This will enable to obtain four directories (« Queen/ », « King/ », ..), each one containing all test images.

To decide if two images are similar, one will compute the number of matches : if there enough matches (threshold to be empirically determined), one considers that the test « unknown » image contains the object of reference. Implement this program, and correctly tune the threshold (do not hesitate to first print the number of matches, to have an idea of the orders of size).

4 Exercice 2 : reconnaissance et localisation / Exercise 2 : localization of recognized objects.

L'objectif est de localiser chaque objet reconnu dans les images. Pour cela, à l'aide des correspondances trouvées, on exploitera maintenant les coordonnées « pixels » des keypoints, en considérant le barycentre. En utilisant opencv, afficher un cercle à l'endroit de ce barycentre, ainsi que le nom de la carte détectée et reconnue.

English : The purpose is to localize each recognized object within test images. For this, based on found matchings between keypoint descriptors, use coordinates of related keypoints to estimate the center of mass of each recognized object and superimposed a circle on the image at this location.