

Reconnaissance d'objets / Object recognition

1 Objectif

L'objectif est d'analyser chacune des images fournies (voir illustration 1) afin d'extraire puis identifier chaque objet. L'identification reviendra à créer une image par objet, sauvegardée au format png dont le nom sera préfixé par sa catégorie (voir tableau ci-après et illustration 4).

Les 4 catégories (ou classes) d'objets sont « Assiette », « Couvert_outil », « Ciseaux » et « Rondelle ». Elles ont les caractéristiques morphologiques et topologiques suivantes:

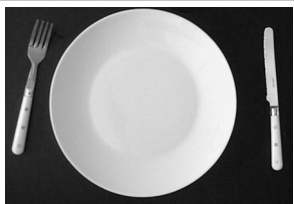
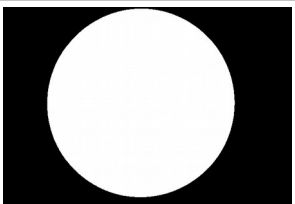
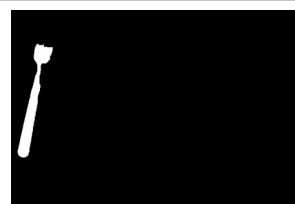

- Assiette, de forme ronde et ne présentant aucune cavité,
- Couverts-outil, de forme allongée et présentant moins de deux cavités,
- Ciseaux, de forme allongée et présentant au moins deux cavités,
- Rondelle, de forme ronde et avec une cavité

Pour atteindre cette objectif, nous allons procéder en 3 étapes : segmentation, extraction des objets et identification (reconnaissance). Pour faciliter la lecture-écriture d'un ensemble d'images placé dans un répertoire donné, un script exemple est fourni.

English : The purpose is to analyse each provided image (see Figure 1) in order to extract and then identify/recognize each objet. Recognition will consists in creating an image per object, saved in png format which a filename corresponding to its identity (see hereafter Table and Figure 4). The 4 object classes are « Plate », « Cutlery-Tool », « Scissors » and « Washer ». Each classe depict following morphological and topological characteristics :

- Plate : curved shape without any cavity
- Cutlery-Tool : elongated with less than two cavities
- Scissors : elongated with at least two cavities
- Washer : curved shape with one cavity

To reach this goal, we will consider 3 steps : segmentation, extraction of objects (connected components saved in different png files) and recognition. An example of script is provided, dedicated to reading/writing a set of image files from/to a directory.

			
Image initiale / Initial image (image_1.png)	Objet de type « assiette » / Plate (assiette_1.png / plate_1.png)	Objet de type «couvert-outil » / « cutlery-tool » (couvert_outil_1.png / tool_1.png)	Objet de type «couvert-outil » (couvert_outil_2.png / tool_2.png)
Résultat attendu pour une image particulière, contenant plusieurs objets de différentes catégories (2 catégories) Example of expected result for a particular image, containing several objects (2 different classes)			

2 Etape 1/Step 1 - Segmentation : Images → Segmentations

La première étape consiste à segmenter l'image. On considérera une approche par région basée sur une simple seuillage dont le seuil sera déterminé par la méthode d'Otsu. On

appliquera une fermeture morphologique (élément structurant de taille 11x11) à l'image seuillée afin de boucher les trous de petite taille. Pour les tests, les images segmentées pourront être sauvegardées dans un répertoire « Segmentation/ » (voir illustration 2).

English : The first step focuses on segmentation. Consider an region-based segmentation approach by thresholding (threshold determined by Otsu). Apply a closing (structure element of size 11x11 pixels) to the thresholded image to fill holes. Save resulting images in a directory named « Segmentation » (see Figure 2).

3 Etape 2/Step 2 - Extraction des objets : Segmentations → Objects

La seconde étape consiste à extraire les objets des images segmentées. Cette étape reposera sur une décomposition de l'image en composantes connexes, en ne préservant que les composantes de taille significative (e.g. surface supérieure à 300 pixels). Pour les tests, on pourra lire les images du répertoire précédent « Segmentation/ » et sauvegarder chaque objet sous forme d'une image dans un répertoire « Objects/ » (voir illustration 3).

English : Extract connected components larger than 300 pixels from segmented images (from the previous «Segmentations/» directory). Save each connected component in Save resulting images in a directory named «Objects/» (see Figure 3).

4 Etape 3 - Reconnaissance des objets : Objects → Recognition

La dernière étape consiste à reconnaître chaque objet. Pour les tests, on pourra lire les images du répertoire précédent « Objects/ » et la sauvegarder sous forme d'une image dans un répertoire « Recognition/ », dont le nom sera préfixé par la classe d'appartenance (voir illustration 4). La première caractéristique concerne la forme générale de l'objet (allongement), que l'on pourra quantifier en considérant la rapport entre la longueur du grand axe et celle du petit axe de l'ellipse englobant la forme (la mesure sera « float(longueur_grand_axe)/float(longueur_petit_axe) »). Pour cette mesure, on utilisera la fonction « *regionprops* » (voir documentation en ligne), de la librairie « scikit-image ». On considérera qu'un objet est « allongé » si ce rapport est supérieur à 2. L'autre caractéristique concerne le nombre de cavités. Il peut être calculé en comptant le nombre de composantes connexes sur l'image de la différence entre l'image binaire dont les « trous » ont été bouchés (fonction scipy « *fill_holes* ») et l'image binaire initiale.

English : Read images from the previous « Objects/ » directory, then recognize objects, and finally save images in the « Recognition/ » directory, with a filename corresponding to identity of the object contained in the image (see Figure 4). For recognition purpose, measure the elongation of each object first : this can be done by computing the ratio between axis («float(major_axis)/float(minor_axis) ») using the « *regionprops* » function implemented by the « scikit-image » library. For classification purposes, one considers that an object is « elongated » if the ratio is larger 2. The other measurement is the number of cavities (or holes), to be computed by counting the number of connected components of the difference image between the « filled » one and the initial binary one (« *fill_holes* » scipy function).

5 Données considérées, résultat intermédiaire et final / Details

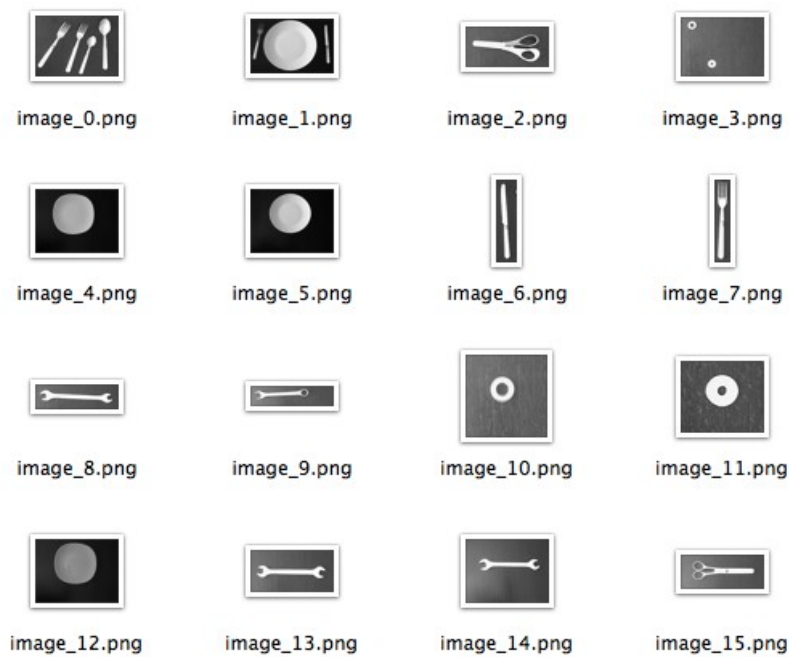


Illustration 1: Données initiales – Initial images

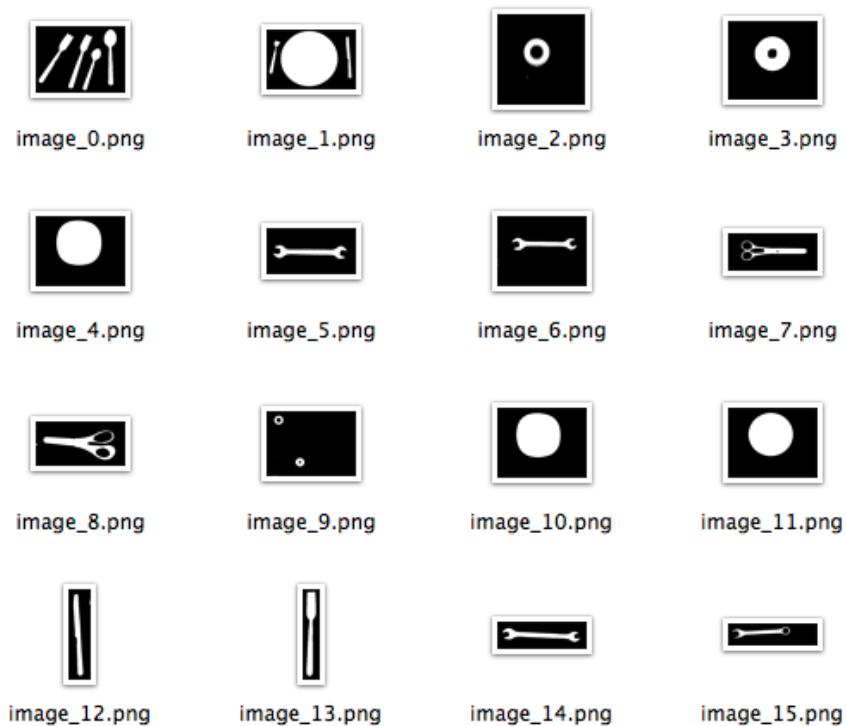


Illustration 2: Données segmentées (étape 1) – Segmentations (step 1)

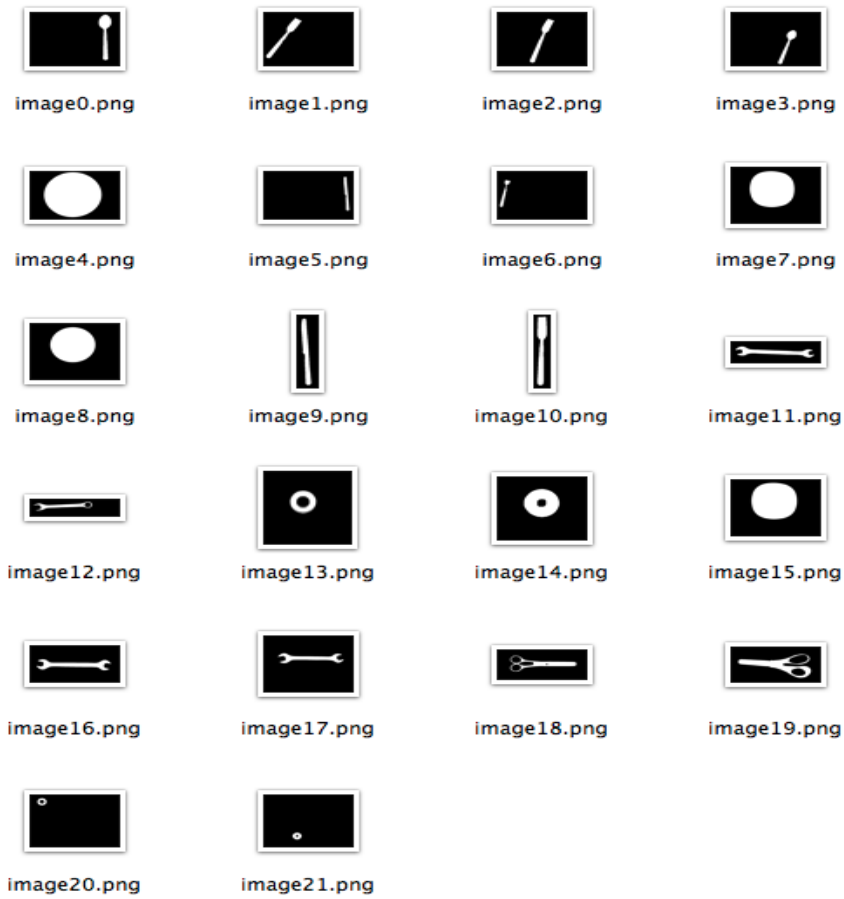


Illustration 3: Objets extraits (étape 2) – Objects (step 2)

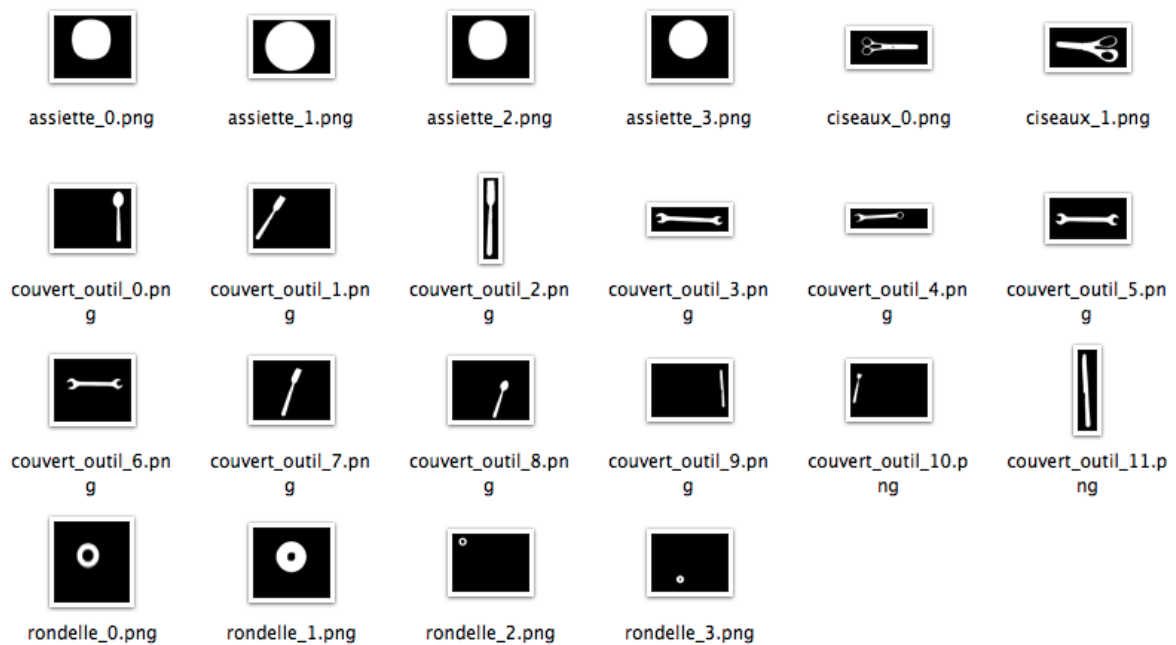


Illustration 4: Objets identifiés (étape 3) – Recognized objects (step 3)

6 Généralisation : apprentissage automatique et arbre décisionnel

Dans le cas précédent, l'identification s'est faite en définissant nous-même les seuils de décision par rapport à l'allongement et au nombre de trous, et en les appliquant ensuite à chaque objet rencontré. Cette étape peut-être automatisée en fournissant, à un algorithme de classification, quelques exemples d'objets pour chacune des classes : cet algorithme estime alors des seuils de décision « raisonnables ». On peut ensuite lui demander de classer les objets inconnus.

Ce principe est illustré par la figure 5 suivante.

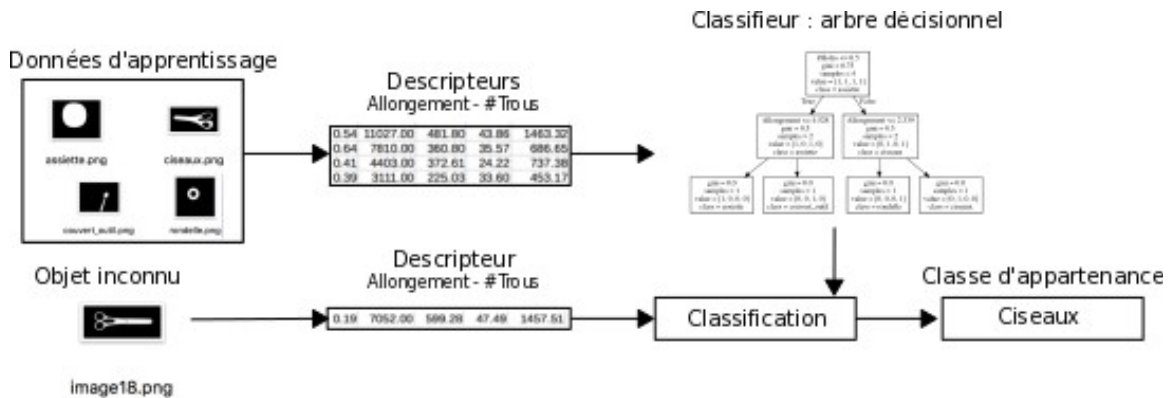


Illustration 5: Apprentissage automatique : exemple appliqué à notre cas, avec un arbre décisionnel

Dans notre cas, nous allons utiliser un arbre décisionnel, implémenté par la célèbre librairie « scikit-learn » dédiée à l'apprentissage automatique (voir figure 6 ci-dessous).

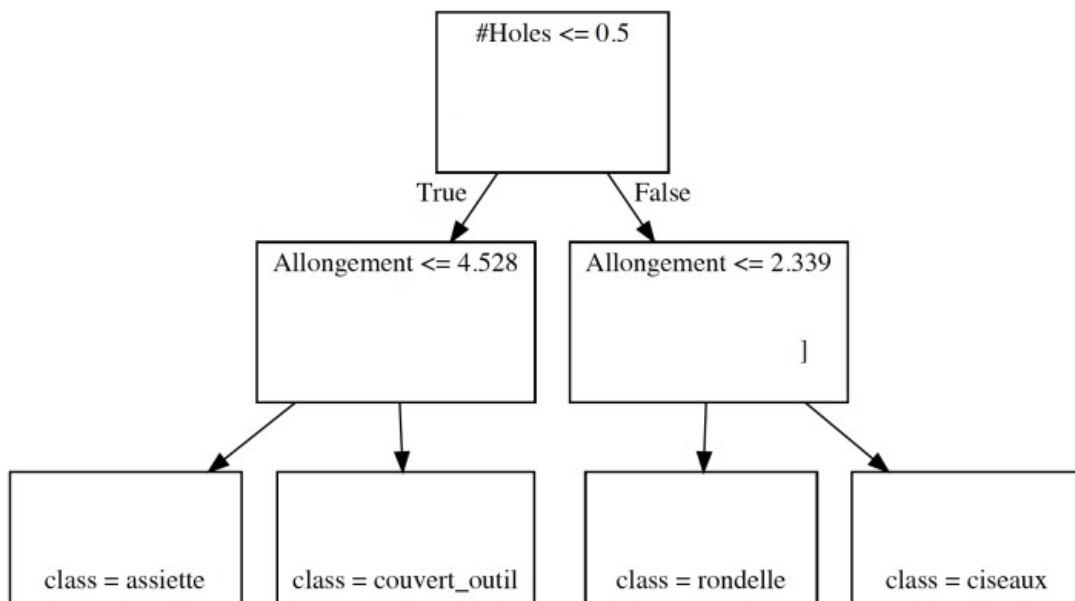


Illustration 6: Exemple d'arbre décisionnel entraîné

Adaptez le programme de reconnaissance précédent en utilisant un arbre décisionnel entraîné sur 4 exemples (un exemple par classe).

On s'inspirera du code suivant : la difficulté concerne le format des tableaux (numpy array) à fournir.

```
#Classe utilisée
from sklearn import tree
clf=tree.DecisionTreeClassifier()

#Apprentissage
clf.fit(features_array,label_array)

# features_array est de la forme
np.array([ [ allongement_objet1, nb_trou_objet1], #e.g. assiette
          .....
          [ allongement_objet4, nb_trou_objet4]])
#Par exemple
[[1.02768907 0.    ] #e.g. assiette
 [3.66928332 2.    ] #e.g. ciseaux
 [8.02771692 0.    ] #e.g. couvert-outil
 [1.00907326 1.    ] #e.g. rondelle
# label_array est de la forme suivante (où, e.g., 0→ « assiette » ; 1→ « rondelle »...)
np.array([ [ 0], #e.g. assiette
          [ 1], #e.g. ciseaux
          [ 2], #e.g. couvert-outil
          [ 3] ]) #e.g. rondelle
#Reconnaissance : label vaudra 0, 1, 2 ou 3 (i.e. « assiette », « rondelle », ....)
label=clf.predict(np.array([[axis_ratio,nb_holes]]))[0]
```

Structure du code à utiliser.