TP2 - Lecture automatique de chiffres par analyse d'image

Consignes

Cadre

Les Tps se font par binôme (ou un monôme en cas de nombre d'étudiants impairs) afin de palier au manque de machines en salle. Cela veut dire qu'il y aura seulement un rendu par groupe.

Rendu

Le rendu se compose des deux éléments suivants:

- Le code complété à partir du répertoire initial qui accompagne ce sujet.
- Un rapport écrit au format word ou pdf selon le modèle disponible dans le repertoiore de travail. Ce rapport doit rendre compte du travail que vous avez effectué en présentant pour chaque question demandé les problèmes à résoudre et la solution proposée avec une explication avec vos propres mots et enfin le résutlat obtenu (par exemple un affichage du résultat obtenu sur plusieurs exemples dans le cas d'une fonction).

Les deux éléments sont **essentiels**. En effet, avoir un code fonctionnel et répondant à l'ensemble des questions. ne suffit pas pour avoir une bonne évaluation. Le rapport permet de rendre compte que vous avez compris ce que vous faisiez.

> *Le dépôt du rendu (code + rapport) se fait sur la plateforme **Github Classroom** dont le lien est disponible sur la page moodle du cours au niveau relatif au TP2. **De plus, il faut aussi déposer uniquement le rapport** sur la zone dépot du TP2 sur la page moodle.*

Test unitaires

Des tests unitaires sont présents dans le dossier **tests** afin de permettre une évaluation rapide de votre progression dans le TP et également de vous permettre de savoir si la solution que vous proposer répond bien à ce qui est demandé en termes de spécifications. Lorsque vous réaliser une fonction, méthode ou classe démandés par une question, il faut exécuter la série de tests unitaires (en éxécutant le fichier de tests) comme présenté dans le dernier TD sur les tests unitaires.

> *Dans le cadre du TP2, deux fichiers concentrent les tests :*
>
> - ***test_Image.py** qui concentre des tests sur une classe **Image** à réaliser.*
> - ***test_reconnaissance.py** qui concentre des tests sur une fonction de reconnaissance de chiffres à écrire.*

Évaluation

L'évaluation prend en compte d'une part le résultat des tests unitaires dans une moindre mesure mais surtout en grande partie la qualité des explications et du rapport. Ce TP est **noté** contrairement au premier TP.

Délai

Afin de vous laisser le temps de faire le rapport si besoin, la date limite du rendu est donnée au **soir du jour où le TP a été programmé (avant minuit)**.

I - Présentation du TP

Représentation des images numériques

Les images numériques sont représentées par des tableaux à 2 dimensions, où chaque élément du tableau correspond à un pixel. Dans le cas d'une image en niveaux de gris, la valeur de chaque élément du tableau correspond à l'intensité du pixel, cette intensité étant généralement codée sur 8 bits, la valeur 0 correspondant au noir et la valeur 255 au blanc et les valeurs intermédiaires correspondant à différents niveaux de gris. Dans le cas d'une image couleur, chaque élément du tableau (c'est-à-dire chaque pixel) est caractérisé par trois grandeurs : une intensité de Rouge, une intensité de Vert et une intensité de Bleu, chacune de ces intensités étant comprises entre 0 et 255.

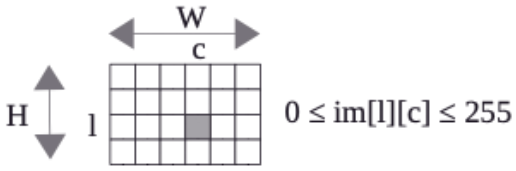


Image en niveaux de gris

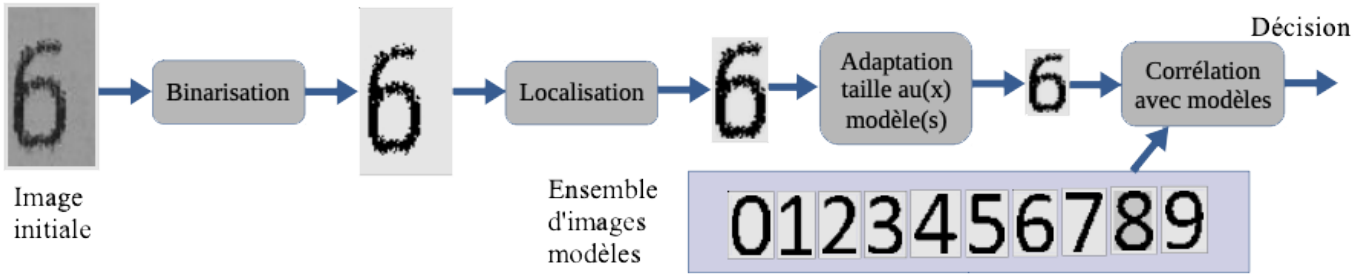
Image RVB couleur

> *Dans la suite de ce TP, nous n'utiliserons que des images en niveaux de gris.*

La reconnaissance automatique de caractères

Aujourd'hui, l'analyse automatique d'images numériques est utilisée dans de très nombreux domaines : en médecine (assistance au geste chirurgical), dans l'industrie (contrôle qualité en bout de chaîne de fabrication), pour la surveillance de la surface du globe (suivi du déplacement des glaciers par imagerie satellitaire), en robotique (aide au déplacement des robots compagnons), en vidéo-surveillance (détection de comportements anormaux dans un lieu public), dans l'accès au savoir (numérisation du fonds des bibliothèques), etc. Parmi toutes les techniques utilisées dans ces différents domaines, la lecture automatique des caractères (OCR : Optical Character Recognition) a maintenant d'excellentes performances, même dans la reconnaissance de l'écriture manuscrite.

L'objectif de ce TP est d'illustrer, parmi les différentes techniques de lecture automatique de chiffres, une des solutions les plus simples : la reconnaissance par corrélation avec des modèles. Le principe de cette approche est détaillé ci-dessous :



- **Étape 1 : binarisation**
  L'image initiale est en niveaux de gris, c'est-à-dire que chaque pixel a une intensité qui est comprise entre 0 (noir) et 255 (blanc). La binarisation transforme l'image en niveaux de gris en une image binaire où les pixels ne peuvent être que **noirs** (intensité = 0) ou **blancs** (intensité = 255). La décision se fait par comparaison à un seuil qui est le paramètre de la méthode et dont le choix peut s'avérer critique. Cette étape permet de simplifier l'image et de faire ressortir l'information utile.
- **Étape 2 : localisation**

  Pour simplifier la recherche par corrélation, on restreint l'image au **rectangle englobant** délimitant le chiffre à reconnaître.
- **Étape 3 : adaptation de la taille au(x) modèle(s)**

Pour que la recherche par corrélation puisse avoir un sens, il faut que les deux images dont on veut mesurer la ressemblance par corrélation soient de la même taille. Il faudra donc **adapter la taille de l'image analysée** à la taille des différents modèles envisagés. Notons que les modèles peuvent avoir des tailles différentes.

- **Étape 4 : mesure de ressemblance par corrélation**

  La mesure de ressemblance par corrélation entre deux images va consister à compter la proportion de pixels de **même intensité** et situés au **même endroit** dans chacune des deux images. Cette mesure va donc varier de 0 à 1. Une valeur égale à 0 signifie qu'aucun des pixels d'une image n'a la même intensité que le pixel correspondant de l'autre image. Une valeur égale à 1 signifie que chaque pixel d'une image a la même intensité que le pixel correspondant de l'autre image (ressemblance parfaite).

- **Étape 5 : Décision.**

  Association de l'image analysée à l'image modèle de **corrélation maximale**.

II - Prise en main de l'environnement

Dans le dossier de travail donné initialement dans ce TP, nous avons l'aborescence suivante:

```
src/
├─ main.py
├─ reconnaissance.py
├─ image.py
tests/
├─ test_Image.py
├─ test_reconnaissance.py
assets/
README.md
```

Le code principal se trouve dans le dossier **src/** pour lequel on a :

- **image.py** : un fichier dans le quel on donne une classe Image qui permet de lire une image (méthode **load**) et de l'afficher (méthode **display**)). un exmeple d'utilisaiton de ces méhtodes est donnée dans le fichier **main.py**
- **reconnaissance.py** : une fichier dans lequel on codera la fonction de reconnaisssance principale.
- **main.py** : le fichier principal d'exécution qui appelle les différentes méthodes/focntions à réaliser afin de voir si elles fontionnent.

À cela s'ajoute les dossiers **tests** qui contient les tests unitaires et **assets** qui continet des images de modèles et de tests.

Dans un premier temps, exécuter le fichier **main.py** et vérifier qu'une image s'affiche bien à l'écran.

> *Dans le cas contraire, il est probable que le chemin pour accéder aux images est mal spécifié. Il faut alors modifier la variable **path_to_assets** en conséquence et finir avec un /.*

Des erreurs s'affichent également sur le terminal mais c'est normal parce que les fonctions à réaliser n'ont pas encore été faites. Vous pouvez commenter la fin du code pour le moment.

III - Travail préparatoire

(1). Analyser attentivement la classe image et remarquer les attributs **H, W** qui indiquent la taille de l'image et l'attribut **pixels** qui contient un tableau 2D numpy contenant les valerus de l'image en pratique.

(2). Écrire une méthode **binarisation(self, S)** à la classe **Image** qui permet de passer d'une image codée sur 256 valeurs à une image avec seulement deux valeurs (0 ou 255). Cela se fait en comparant pour chaque pixel de l'image la valeur du pixel à une valeur choisie par l'utilisateur (entrée **S** de la méthode). Ainsi en pratique, il faut itérer sur tous les pixels du tableau numpy et comparer la valeur au seuil. On veut également que le résultat soit donné sous la forme d'une nouvelle image que l'on crée dans la fonction afin de ne pas mmodifier l'image de base.

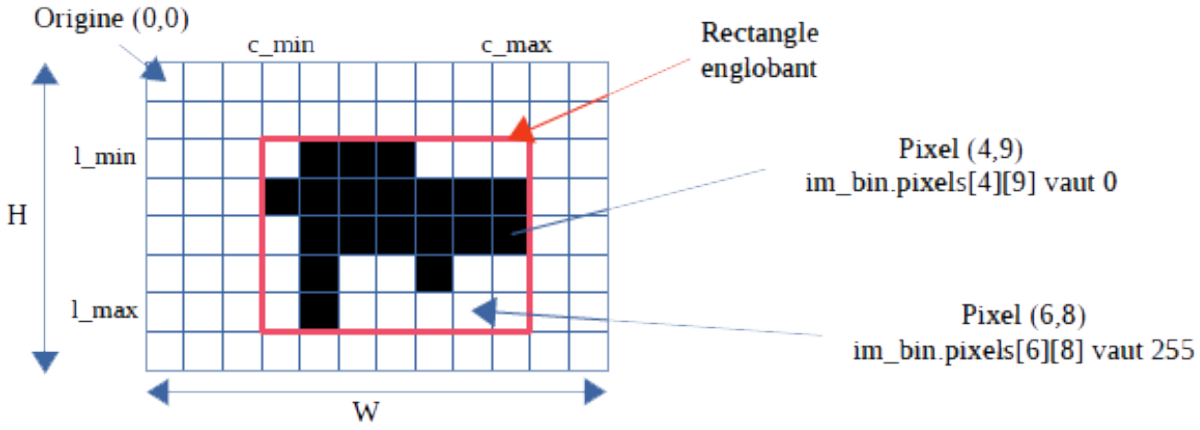
Pour cela on vous donne le botu de code suivant à compléter:

```python
    def binaris(self, S):
        # creation d'une image vide
```

```
        im_bin = Image()

        # affectation a l'image im_bin d'un tableau de pixels de meme taille
        # que self dont les intensites, de type uint8 (8bits non signes),
        # sont mises a 0
        im_bin.set_pixels(np.zeros((self.H, self.W), dtype=np.uint8))

        # TODO: boucle imbriquees pour parcourir tous les pixels de l'image im_bin
        # et calculer l'image binaire


        return im_bin
```

> *Il est également possible de procéder par un masque sur les indices de valeurs comme dans la dernière fonction du TP précédent. Pour plus de détails, voir: https://flotpython-primer.readthedocs.io/fr/latest/2-07-numpy-mask.html*

(3). Écrire une méthode **localisation(self)** à la classe **Image** calculant et retournant l'image recadrée sur le chiffre à identifier. Le principe de ce recadrage consiste, sur une image binaire **im_bin** de dimension $H \times W$, à déterminer les coordonnées l_{min}, l_{max}, c_{min} et c_{max} du rectangle englobant la forme noire (valeurs 0) et à construire l'image limitée à ce rectangle englobant.



> *En python ima.pixels[20:40,30:55] désigne la partie de l'image ima, limitée aux lignes 20 à 39 et aux colonnes 30 à 54.*

IV - Reconnaissance automatique de chiffre

(1). Essayer de lancer le fichier **main.py** et de voir le résultat de la méthode de binarisation. Essayer avec différetnes valeurs de seuil et observez le résultat.

Lancer également le fichier de tests **test_Image.py** pour voir si votre méthode répond bien aux spécifications demandées (regarder seulement le résultat des tests en rapport avec la binarisation). Corriger au besoin en analysant les tests qui ne réussisent pas ou pour lequel il y a des erreurs.

(2). Essayer de lancer le fichier **main.py** et de voir le résultat de la méthode de localisation. Essayer avec différetnes valeurs de seuil et observez le résultat.

Lancer également le fichier de tests **test_Image.py** pour voir si votre méthode répond bien aux spécifications demandées. (regarder seulement le résultat des tests en rapport avec la localisation). . Corriger au besoin en analysant les tests qui ne réussisent pas ou pour lequel il y a des erreurs.

(3). On dispose de la fonction **resize** de la libraire **skimage** permettant de donner à une image des dimensions imposées. Cette fonction s'utilise de la manière suivante :

```
resize(im.pixels, (new_H,new_W), 0)
```

où:

- 'im' est l'image dont on veut modifier les dimensions,
- 'new_H' et 'new_W' sont les nouvelles dimensions de l'image,
- '0' indique l'absence d'interpolation ce qui permet de conserver une image binaire si l'image de départ est binaire,
- 'im_resized' est l'image redimensionnée

Ajouter à la classe **Image** la méthode **resize(self,new_H,new_W)** qui redimensionne l'image à la taille voulue et renvoie un autre objet de type **Image** en sortie. Tester cette méthode dans le fichier **main.py** sur l'image obtenue par l'étape de localisation. Pour ce test, on choisira des dimensions quelconques, (60,100) par exemple.

> *Le résultat de la fonction resize est codé sur une échelle en float et non plus en int comme précédemment dans le TP. Il faut ainsi faire une conversion pour se ramener à notre cas. Si **pixels_resized** est un tableau numpy 2D résultant de la fonction resize, alors on peut se ramener à une image en int avec la commande suivante:*
>
> ```
> np.uint8(pixels_resized*255)
> ```

Lancer également le fichier de tests **test_Image.py** pour voir si votre méthode répond bien aux spécifications demandées. (regarder seulement le résultat des tests en rapport avec la localisation). . Corriger au besoin en analysant les tests qui ne réussissent pas ou pour lequel il y a des erreurs.

(4). Ajouter à la classe **Image**, la méthode **similitude(self, image)** qui mesure la similitude par corrélation d'images entre l'image représenté par l'objet courant (**self**) et un objet de type **Image** entrée en paramètre. La procédure pour le calcul de similitude est décrite dans la présentation du TP.

(5). Dans le fichier **reconnaissance.py**, écrire la fonction **reconnaissance_chiffre(image, liste_modeles, S)** qui va effectuer la reconnaisance de chiffre sur l'image **image** donnée en entrée de la fonction. Pour cela il faudra dans la fonction tout à tour, binariser l'image et la localiser. Ensuite, il faut calculer sa simiitude à tous les modèles (en redimensionnant l'image à la taille du modèle) et trouver le modèle pour lequel il y a la plus grande similitude. Cela peut être fait en parcourant une liste d'images modèles et en sauvegardant la simimiltude maximale ainsi que l'indice de l'image avec la similitude maximale. La fonction doit renvoyer un entier compris entre 0 et 9.

Un autre paramètre de la fonction est **liste_modeles** qui est une liste d'objets de type **Image** correspondant aux modèles des deix chiffres. Elle est obtenue à l'aide de la fonction **lecture_modeles(chemin_dossier)** qui va lire les fichiers images contenus dans **assets/**, lorsque le bon chemin est donné en entré, et renvoyer une liste d'objets **Image**. Un exemple est donné dans le fichier **main.py**

Le dernier paramètre est le seuil de binarisation.

Lancer également le fichier de tests **test_reconnaissance.py** pour voir si votre méthode répond bien aux spécifications demandées. Corriger au besoin en analysant les tests qui ne réussissent pas ou pour lequel il y a des erreurs.

(6). Essayer la fonction de reconnaissance en modifiant l'image de test dans **main.py** avec différentes images disponibles dans **assets/** et en modifiant également le seuil avec quelques valeurs (pas besoin d'en faire 100...). Présenter les résultats dans le rapport sous forme de tableau (différentes images en ligne et différents seuil en colonne). Proposer une valeur de seuil qui marche le mieux selon vos expérimentations.