


TP 1 : Installation d'un espace de travail et manipulation d'images

 = Cet item indique qu'on attend une réponse écrite dans le Readme.md, des points sont associés à la réponse.

Je rends un code propre organisé de façon cohérente. Le « clean code » fait partie de la notation.


Rendu = lien vers votre repo git donc l'accès m'est autorisé.

Partie A : Mise en place

J'ai un environnement de code fonctionnel. Checklist :

- ☐ Je me place dans un environnement virtuel
- ☐ Je peux exécuter un script python via la ligne de commande
- ☐ Je peux exécuter un script python via l'interface de mon IDE
- ☐ Je sais utiliser le debugger
- ☐ Je sais créer et utiliser des cellules interactives
- ☐ J'ai installé les packages suivants : numpy, opencv, matplotlib, sklearn
- ☐ J'ai un compte github ou gitlab
- ☐ J'ai initialisé un repo git en local et en remote
- ☐ Je suis capable de pull et push avec git

Etapes

1. Créer un nouveau dossier avec un fichier python et un readme.md
2. Mettre en place un repo git en local
3. Mettre en place le repo git distant (remote)
4. Créer un environnement virtuel (avec pyenv, ou si maîtrisé conda)
5. Se placer dans l'environnement virtuel
6. Installer les packages suivants :
 - a. Numpy
 - b. Sklearn
 - c. Matplotlib
 - d. Opencv
7. Je manipule numpy
 - Créer une array numpy, X, de 1000 points avec valeur aléatoire dans l'intervalle [0, 3]
 - Calculer la moyenne, l'écart type et la médiane de cette liste. Arrondir dans le code les valeurs au centième. Noter les valeurs 

- Créer une array numpy, `X_bis`, de 1000 points avec valeur aléatoire dans l'intervalle `[0, 3]`
 - Calculer la moyenne, l'écart type et la médiane de cette nouvelle liste. Noter les valeurs
 - Comparer les résultats de moyenne, écart type et médiane des listes `X` et `X_bis`.
 - Fixer l'aléa (seed) pour pouvoir reproduire les résultats et vérifier que la cellule donne maintenant des valeurs constantes. Pour la suite, on se rappellera que les modèles d'apprentissage machine ne sont pas déterministes, il est donc important de fixer l'aléa pour avoir des résultats reproductibles.
 - A quoi sert le fait de fixer l'aléa (seed) ?
 - Créer une liste, `y`, de 1000 points ayant la valeur de $\sin(X)$ auquel on ajoute un bruit gaussien aléatoire ayant une amplitude de 10% (0.1).
8. Je manipule matplotlib
- Visualiser `y` en fonction de `X` sous forme de graph 'scatter'
- Pour la suite, on oubliera pas d'utiliser `plt.show()` après chaque plot matplotlib.
- Changer la taille de la figure. Quelle ligne de code permet de le faire ?
 - Visualiser le bruit gaussien, `noise`, sous forme d'histogramme. Le nombre de bins est fixé à 50.
 - A quelle fonction la distribution de `noise` fait penser ?
9. Je manipule opencv
- Je lis l'image 'image_1.jpg' et je la met dans la variable `img`. Quelle méthode est utilisée ?
 - Je visualise l'image dans le bon espace couleur i.e. je vois la même chose que lorsque j'ouvre l'image jpg. Je ferme l'image (avec 'entrée' et non avec la croix de la fenêtre).
 - Afficher l'image en noir et blanc. Que faut-il ajouter à la ligne de code précédente ?
 - Créer une méthode 'display_image' qui prend en argument une image `img` et de façon optionnelle un nom `name` (str) de fenêtre et qui affiche l'image avec opencv. Cette méthode sera mise dans un fichier `utils.py`.
 - Importer et appeler la méthode 'display_image' précédemment créée pour afficher l'image 'image_1.jpg'. On utilisera cette méthode dans la partie B.






Partie B : Manipulation d'image

Dans cette partie, on travaille avec l'image 'image_1.jpg'.






Chaque pixel d'une image est entre 0 et 255. Quelle couleur est représentée par le 0 ?
Quelle couleur est représentée par le 255.

1. Seuillage

Je cherche à segmenter le chat blanc de l'image.

- Pour cela, je commence par la plus simple des techniques : le seuillage manuel. Expliquer ce terme. 
- Appliquer un seuillage (threshold) en déterminant les seuils minimum et maximum des pixels que je veux garder.
- Quels sont les seuils min et max qui optimisent la segmentation ?
- Enregistrer l'image en noir et blanc, l'intégrer au Readme .
- J'utilise maintenant le seuillage d'OTSU. Appliquer le seuillage d'OTSU à chacun des canaux de l'image.
- Quelles sont les seuils optimaux trouvés par ce seuillage pour chacun des canaux hsv de l'images ? 
- Avec opencv, il est possible de fusionner des images pixel par pixel. Cela se fait par des opérations logiques OR et AND avec des méthodes telles que : `cv2.bitwise_or`. Fusionner des images provenant des canaux hsv du seuillage d'OTSU précédemment obtenu.
- Optimiser la segmentation du chat blanc par de telles opérations. Enregistrer l'image en noir et blanc, l'intégrer au Readme .
- Comparer les deux techniques de seuillage utilisées .


2. Détection de contour

- Appliquer le filtre de Sobel pour détecter les contours du clavier. Expliquer à quoi serve chacun des paramètres et noter les valeurs optimales pour repérer le clavier .
- Appliquer le filtre de Sobel pour détecter le contour du chat blanc. Noter les valeurs optimales des paramètres pour cela .
- Appliquer le filtre de Canny pour détecter les contours du clavier. Expliquer à quoi serve chacun des paramètres et noter les valeurs optimales pour repérer le clavier .
- Appliquer le filtre de Canny pour détecter le contour du chat blanc. Noter les valeurs optimales des paramètres pour cela .
- Comparer les deux détections de contour dans le repérage du chat blanc .

3. Transformation par point

Appliquer les transformations de valeur par point suivantes : color jitter, inversion, noir et blanc, augmentation & diminution de luminosité, augmentation & diminution de contraste.

Jouer avec les paramètres et sélectionner 7 transformations qui maximisent la diversité de l'image tout en restant des images crédibles d'observation.

Afficher dans un unique plot matplotlib les 7 images transformées et l'originale dans une grille de 8*2. Enregistrer la figure et l'intégrer aux réponses du Readme .

Appliquer les transformations de position par point suivantes : recadrage (cropping), retournement horizontal & vertical, rotation, déformation, mise en perspective.

Jouer avec les paramètres et sélectionner 7 transformations qui maximisent la diversité de l'image tout en restant des images crédibles d'observation.

Afficher dans un unique plot matplotlib les 7 images transformées et l'originale dans une grille de 8*2. Enregistrer la figure et l'intégrer aux réponses du Readme



4. Distance entre images

Créer une méthode 'image_euclidean_distance' à placer dans le fichier utils.py, qui prend en entrée deux images opencv, et qui ressort la distance euclidienne entre les deux images. La distance vaut 0 si les images sont les mêmes.