TP 01 Bases du traitement d'images

A. B., CERIST

03/11/2024

1 Introduction

Pour commencer : téléchargez les fichiers sources sur le dossier TP 01 à partir de https://github.com/DSARU-CERIST/FormationCERIST/tree/main/BTI_main_0. Pour exécuter le projet localement, vous aurez besoin d'installer IPython/Jupyter, par exemple, voir les instructions à https://docs.jupyter.org/en/latest/install.html. Les notebooks sont écrits pour Python 3.x. Lancez Jupyter et ouvrez starter.ipynb. Alternativement, vous pouvez importer la version autonome du notebook dans Colaboratory (https://colab.research.google.com/) et l'exécuter sans rien installer. Utilisez File->Upload Notebook dans Colab et ouvrez le notebook starter.ipynb. Vous pouvez aussi cloner le dossier comme suit :

```
!git clone https://github.com/DSARU-CERIST/BTI.git 'chemin_de_sauvegarde'
```

Listing 1: Cloner un repository git distant dans un répertoire local.

Si vous n'êtes pas déjà familiarisé avec un environnement Python/Numpy, il est recommandé de faire une introduction telle que : http://cs231n.github.io/python-numpy-tutorial. Si vous travaillez sur Google Colaboratory, un disque temporaire vous est fourni pour les opérations E/S. Il est possible d'enregistrer votre travail de façon permanente en vous connectant au disque associé à votre compte Google en utilisant les commandes suivantes :

```
from google.colab import drive
drive.mount('/content/drive')
```

Listing 2: Montage de Google Drive sur Google Colaboratory.

Ne pas oublier de se positionner dans le répertoire principal. Vous serez amenés à travailler sur le fichier starter.ipynb. Les fonctions développées durant ce TP seront sauvegardées dans le fichier my_functions.py.

2 Partie I: Lecture/Ecriture d'images sous Python

La méthode *imread* de Matplotlib prend en paramètre le chemin d'un fichier à partir du répertoire courant de travail. Ce dernier peut être perçu à gauche de votre notebook. Les formats acceptés par la fonction sont : BMP, JPEG, GIF, PNG, TIFF, ... Dans le cas d'une image en niveaux de gris, la matrice renvoyée par la fonction est une matrice d'entiers (entre 0 et 255) de dimension 2. Dans le cas d'une image en couleurs c'est une matrice de dimension 3.

```
I = plt.imread('chemin_vers_image')
(I.shape) # Afficher sa taille
```

Listing 3: Lecture image en niveaux de gris

Lire une image en couleurs à partir du dossier figs et afficher sa taille en utilisant les mêmes instructions. N.B : L'appel de la méthode *imread* renverra un tableau numpy uint8 (entier non signé). Pour vérifier le type d'une variable on utilise la fonction *type(I)* et pour connaitre les types de données utilisées par notre variable on utilise la commande *I.dtype*. Les calculs effectués à partir des valeurs de I peuvent alors poser

des problèmes liés aux opérations sur les entiers. Pour effectuer des calculs en nombres réels, effectuez la conversion suivante : $I = I.astype(np.float32)$.

La fonction *plt.imsave('chemin.extension',matrice)* prend comme arguments le chemin de l'image à créer ainsi que son extension et la matrice à partir de laquelle l'image sera créé. Cette instruction convertit la matrice en une image au format désiré dans le répertoire préalablement défini. Veillez à convertir la matrice en données de type entier avant de l'enregistrer $I = I.astype(np.uint8)$ Pour visualiser l'image enregistrée, veillez d'abord à lire l'image comme vu plus haut, une fois que la matrice équivalente est obtenue, la commande pour afficher la matrice sous forme d'image est la suivante : *plt.imshow(I)*

Pour visualiser la surface d'une image (en utilisant le fichier view surface.py) on peut faire comme décrit ci-dessous :

```
from view_surface import * # importera toutes les fonctions du fichier view_surface.py
view_surface('chemin_vers_image_ngris')
```

Listing 4: Importer une fonction contenue dans un fichier .py à partir d'un notebook.

Python supporte des outils très puissants en matière de traitement d'images. Les images peuvent être traitées à l'aide de différentes bibliothèques comme ImageIO, OpenCV, Matplotlib, PIL, etc.

3 Partie II: Manipulation d'images sous Python

Durant cette partie du TP, vous serez amenés à écrire une série de fonctions en utilisant ce que vous aurez appris dans la partie I. Le but étant de vous familiariser avec les différentes notions de traitement et de pré-traitement d'images.

En utilisant les images à votre disposition écrire et afficher le résultat d'une fonction :

- **OpenImage** : ayant un unique argument et renvoyant trois variables (la matrice équivalente I, le nombre de lignes de la matrice L et le nombre de colonnes C).
- **Divide** : Ecrire une fonction renvoyant en sortie les trois canaux séparés d'une image.
- **HSV** : Ecrire une fonction renvoyant le format HSV d'une image.

```
import cv2
image = cv2.imread("chemin_de_l'image", 3)
image_HSV = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
```

Listing 5: Format HSV en utilisant OpenCV sous python.

- **CountPix** : ayant un argument et renvoyant une variable (N le nombre de pixels de l'image).
- **FactPix**: ayant trois arguments (deux étant des variables que vous choisirez librement pour modifier les intensités de l'image) et renvoyant une variable (I-fact étant l'image modifiée).
- **Func-a** : Ecrire une fonction renvoyant en sortie, le logarithme, l'exponentielle, le carré et la racine de l'image en entrée.
- **Func-m** Ecrire une fonction renvoyant la moyenne μ et la déviation standard $\sqrt{\frac{\sum (x_i - \mu)^2}{N}}$ d'une image.
- **Normalize** : ayant trois arguments et renvoyant une variable en sortie (Inorm étant l'image normalisée i.e. modification de l'intervalle de variation des niveaux de gris de l'image).
- **Inverse** : ayant un argument et renvoyant une variable en sortie (Inv étant l'image inverse ou communément appelé le mode négatif i.e. $Inv = \max(I) - I$).
- **CalcHist** : ayant un argument et deux variables en sortie (H étant un vecteur représentant l'histogramme de I, et b un vecteur représentant les groupes d'intervalles égaux appelés aussi bins).

- **Threshold** : ayant deux arguments et une variable en sortie (T étant l'image obtenu après seuillage i.e. valeurs supérieurs à un seuil déterminé en argument sont mis à 255 et le reste des pixels à 0).
- **Func-j** : Une fonction définie par une série d'instructions (à partir des fonctions déjà écrites) pour pouvoir, à partir du chemin d'une image, l'ouvrir, l'afficher, afficher son histogramme, l'inverser, calculer son histogramme après inversion et visualiser le résultat des traitements.
- **Func-t** : Une fonction définie par une série d'instructions pour pouvoir, à partir du chemin d'une image, l'ouvrir, l'afficher, afficher son histogramme, la normaliser sur l'intervalle [10, 50], calculer son histogramme après normalisation et visualiser le résultat des traitements. Interpréter le résultat de la visualisation.
- **Func-f** : Une fonction définie par une série d'instructions pour pouvoir, à partir du chemin d'une image, l'ouvrir, l'afficher, afficher son histogramme, la seuiller par un seuil s = 128, calculer son histogramme après seuillage et visualiser le résultat des traitements.

4 Partie III: Exercice

Créer la matrice I définissant l'image ci-dessous :

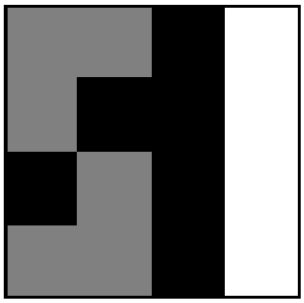

Figure 1: Indiçage d'intensités de pixels

Où un pixel noir a la valeur 0, un pixel gris la valeur 128 et un pixel blanc la valeur 255.

La visualiser avec *plt.imshow* et regarder son histogramme en usant de la fonction CalcHist(I) générée dans la partie II. Pour créer une matrice de nombres aléatoires, il est possible d'utiliser les méthodes *Numpy* suivantes :

```
import numpy as np

ran_d = np.random.rand(9,9)
ran_dN = np.random.randn(7x7)
ran_dINT = np.random.randint(low = 0, high = 8, size = (8,8))
```

Listing 6: Génération de matrices de valeurs aléatoires sous Numpy.

La méthode *rand* génère dans ce cas-ci une matrice de taille (9x9) de nombres aleatoires compris entre 0 et 1. *randn* permet dans ce cas de créer une matrice de taille (7x7) dont les valeurs suivent une distribution normale standard. La methode *randint* génère dans ce cas-ci une matrice de taille (8x8) dont les valeurs sont supérieures ou égaux à "low" et strictement inferieures à "high". Utilisez ces fonctions pour générer une matrice de taille 512×512. Pour randn, testez en utilisant une moyenne de 128 et un écart-type variable aléatoirement généré. Visualisez l'image obtenue et son histogramme. Que remarquez-vous sur cet histogramme ?