Introduction au traitement d'images avec `Python`

Lycée Richelieu
Rueil Malmaison

mars 2025

L'objectif de cette séance est de répondre aux questions en complétant le script présent dans le même répertoire que ce sujet. Si la question requiert de rédiger en langage naturel, utiliser les commentaires pour répondre.

À la fin, on déposera le script dans ce répertoire en ligne avec le mot de passe *TP_images_2025* et on veillera à ce que le nom du fichier permette d'identifier le candidat. Par exemple `script_images.py` n'est pas explicite tandis que `script_images_pa_fournie.py` l'est.

On rappelle que le script doit pouvoir être exécuté sans erreur de compilation par l'examinateur. Si certaines des fonctions conduisent à des erreurs, il faut commenter les lignes correspondantes.

1 Manipulation basique d'images

1.1 Introduction

Pour simplifier, en informatique, il existe deux manières de représenter une image selon qu'on la considère comme

- une grille de points de couleurs appelés pixels ;
- un ensemble de courbes et de surfaces que l'on peut définir à partir de courbes et surfaces élémentaires (cercles, ellipses, courbes polynomiales...).

Dans le premier cas, on parlera d'images matricielles (bitmap en anglais) et dans le second cas, on parlera d'images vectorielles.

Un test très simple qui permet de savoir si l'on se trouve face à une image matricielle consiste à zoomer sur un point. Au bout d'un moment, on perçoit des carrés qui correspondent aux pixels. On appelle cet effet « pixélisation. »

Selon les usages on privilégiera l'un ou l'autre de ces formats. L'avantage du format matriciel, c'est qu'il peut être utilisé pour tous les types d'images tandis que le format vectoriel requiert d'identifier dans l'image des contours et des surfaces. En revanche, le format vectoriel est beaucoup plus sobre en terme d'utilisation de mémoire et il présente aussi le gros avantages de ne pas « pixéliser » l'image lorsque l'on zoome.

En pratique, les appareils photos et les scanners enregistrent au format matriciel tandis que ce document est codé en format vectoriel [1].

Dans toute la suite, nous allons nous intéresser aux images en format matriciel. Nos algorithmes seront en général conçus pour :

- ouvrir une image avec `Python` et la convertir en tableau correspondant aux pixels ;
- travailler sur le tableau pour obtenir l'effet recherché ;
- enregistrer l'image modifiée avec `Python`.

1. La plupart des pdfs sont codés en format vectoriel

Le module `PIL` [2] sera privilégié.

Avant d'aller plus loin, il convient de se document un peu.

(Étape 1) Lire la page Wikipedia consacrée aux images matricielles, en particulier le paragraphe *caractéristiques*.

Pour rendre ce TP plus abordable, nous allons uniquement manipuler l'image en noir en blanc que vous avez choisi d'apporter. En pratique, la couleur de chaque pixel est codée, sur un octet, par un niveau de gris qui est un entier compris entre 0 (noir) et 255 (blanc).

1.2 Ouvrir une image et modifier quelques pixels

Le script *manipulation_images.py* disponible dans le même répertoire que le sujet propose l'ouverture d'une image, l'affichage de sa taille ainsi que la manipulation de quelques pixels.

(Étape 2) Ouvrir le fichier *manipulation_images.py*, modifier les deux lignes correspondant aux chemins afin qu'ils désignent les bonnes images puis exécuter le script.

(Étape 3) Ouvrir l'image modifiée obtenue afin d'analyser le script.

- *Algorithme n° 0:* Faire un code qui ouvre votre image d'origine, la transforme en tableau puis crée un carré noir de taille 10×10 au centre de l'image et enfin enregistre cette nouvelle image avec un nom différent. *Répondre dans la section du code prévue à cet effet.*
- *Algorithme n° 1:* Reprendre le script précédent de manière à ce qu'il affiche maintenant un disque blanc de rayon 5 centré au milieu de l'image.

Il y a deux erreurs classiques que l'on peut faire en manipulant des tableaux : on peut essayer d'accéder à une case en dehors du tableau, c'est l'erreur *index out of range*; ou bien on peut essayer d'accéder à une coordonnée non entière.

2 Création d'effets sur les images

L'objectif de cette question est de recoder « à la main » quelques effets courants des logiciels de traitement d'image.

Il ne s'agit pas d'exploiter les fonctions préexistantes du module `PIL` mais bien de recoder ces effets en manipulant l'image pixel par pixel. Il est donc indispensable de prendre quelques notes et de faire des calculs sur une feuille.

2.1 Effet de seuil et de négatif

- *Algorithme n° 2:* Faire une fonction appelée `seuil` qui :
  - prend en entrée un nom d'image à modifier, un seuil de niveau de gris, un nom d'image de sortie ;
  - transforme en noir les pixels de l'image à modifier dont le niveau de gris est supérieur au seuil et transforme en blanc les autres pixels ;
  - enregistre l'image obtenue avec le nom spécifié pour l'image de sortie.
- *Algorithme n° 3:* Faire une fonction appelée `negatif` qui :
  - prend en entrée un nom d'image à modifier, un nom d'image de sortie ;
  - renverse les niveaux de gris (comme sur le négatif d'une photo argentique) ;
  - enregistre l'image obtenue avec le nom spécifié pour l'image de sortie.

2. Python Image Library

2.2 Transformations géométriques

Pour aborder les deux prochaines questions, il faut se demander, dans le cas des symétries et des rotations, quelles sont les coordonnées de l'image d'un pixel de coordonnées (n, p).

2.2.1 Miroir vertical

On pourra s'inspirer de l'exemple de fonction qui crée un miroir horizontal pour fabriquer les algorithmes suivants.

- *Algorithme n° 4:* Faire une fonction appelée `miroir_vertical` qui :
  - prend en entrée un nom d'image à modifier, un nom d'image de sortie ;
  - applique un effet miroir vertical à l'image, c'est à dire une symétrie par rapport à l'axe vertical passant par le centre de l'image ;
  - enregistre l'image obtenue avec le nom spécifié pour l'image de sortie.

2.2.2 Rotation de +90°

- *Algorithme n° 5:* Faire une fonction appelée `rotation_90` qui :
  - prend en entrée un nom d'image à modifier, un nom d'image de sortie ;
  - applique une rotation de 90° à l'image ;
  - enregistre l'image obtenue avec le nom spécifié pour l'image de sortie.

2.2.3 Réduction de x%

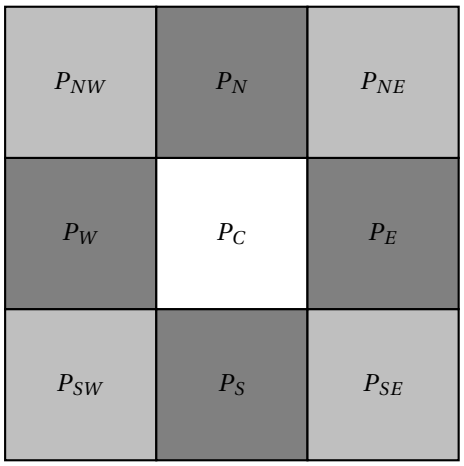
- *Algorithme n° 6:* Faire une fonction appelée `reduction` qui :
  - prend en entrée un nom d'image à modifier, un pourcentage x (un entier compris entre 1 et 99), un nom d'image de sortie ;
  - crée une image réduite dont les longueurs et largeurs sont x% des longueurs et largeurs d'origine ;
  - enregistre l'image obtenue avec le nom spécifié pour l'image de sortie.

On sera particulièrement attentif aux erreurs liées aux coordonnées de pixel non entières ! En particulier, on pourra exploiter la fonction `int` qui convertit un flottant en entier.

2.3 Flou

L'idée pour créer un flou c'est de remplacer le niveau de gris d'un pixel par une moyenne des niveaux de gris des pixels environnants. Plus la moyenne porte sur une zone étendue autour du pixel à modifier et plus le flou est intense.

Au départ, on fera un flou très simple qui remplace chaque niveau de gris par la moyenne des huit pixels les plus proches représentés et nommés dans le dessin ci-dessous.



On choisira un poids de 1 pour les quatre pixels adjacents et un poids de $\frac{1}{2}$ pour chacun des pixels diagonaux. Ainsi, la valeur du pixel central sera donnée par le calcul suivant puis arrondie en entier :

$$P_C = \frac{P_N + P_W + P_S + P_E + \frac{1}{2}(P_{NW} + P_{SW} + P_{SE} + P_{NE})}{6} = \frac{2(P_N + P_W + P_S + P_E) + P_{NW} + P_{SW} + P_{SE} + P_{NE}}{12}$$

Algorithme n° 7: Faire une fonction appelée `flou` qui :

- prend en entrée un nom d'image à modifier, un nom d'image de sortie ;
- applique le flou défini plus haut ;
- enregistre l'image obtenue avec le nom spécifié pour l'image de sortie.

> ⚠ On sera attentif au calcul du flou pour les pixels situés « au bord. » C'est une zone où l'on risque en effet de sortir du cadre. Une mesure prudente consiste donc à ne pas créer de flou pour ces pixels.

2.4 Détection de contours

Un contour peut se définir comme le lieu où se produit un changement brutal des valeurs des pixels. On peut ainsi détecter des contours horizontaux, verticaux, diagonaux selon la direction du changement observé.

Voici quelques principes qui nous permettrons de détecter et d'afficher les contours :

- les contours seront affichés en noir sur une image blanche de même taille que l'image en entrée ;
- pour mesurer un changement dans les valeurs de pixels, on établira un indicateur en effectuant un calcul similaire au flou ;
- en pratique un changement brutal de valeur de pixels sera un booléen qui prendra pour valeur *vrai* lorsque la différence absolue entre les valeurs des pixels voisins dépassera un certain seuil.

En reprenant la terminologie de pixels établie dans le paragraphe précédente, la formule suivante mesure un contour vertical (donc une variation horizontale)

$$\Delta_V = (|P_{NE} + 2P_E + P_{SE} - (P_{NW} + 2P_W + P_{SW})| \geq s) \text{ avec } s \text{ le seuil}$$

On notera que Δ_V est un booléen. Lorsque Δ_V sera vrai, il suffira donc de remplacer les pixels P_N, P_C, P_S par la valeur correspondant au noir dans l'image blanche de sortie.

- *Algorithme n° 8:* Faire une fonction appelée `contoursv` qui :
  - prend en entrée un nom d'image où l'on veut détecter les contours, un nom d'image de sortie, une valeur de seuil ;
  - crée une image noir et blanc qui affiche en noir les contours verticaux et en blanc le reste, à partir de la fonction Δ_V définie plus haut ;
  - enregistre l'image obtenue avec le nom spécifié pour l'image de sortie.

On pourra tester l'algorithme sur différentes valeurs de seuil.

- *Algorithme n° 9:* Dupliquer et améliorer la fonction précédente pour qu'elle détecte et trace à la fois les contours verticaux et horizontaux.

2.5 Effet créatif

- *Algorithme n° 10:* Faire une fonction appelée `mon_effet_a_moi` qui crée sur une image un effet créatif, de préférence non connu des logiciels de traitement d'images.