TD Manipulation d'Images en niveau de gris : Tramage.

Ce sujet de TD s'inspire très largement d'un sujet de la célèbre épreuve écrite PC 2011 "d'informatique avant l'informatique".

Image A

Les ordinateurs et de nombreux dispositifs électroniques (caméras numériques, écrans, etc.) représentent les images comme des matrices de nombres entiers. Dans ce problème, on se limite aux images rectangulaires en teintes de gris.

En machine, une telle image est la donnée d'une hauteur H, d'une largeur L, et d'une matrice M d'entiers de H lignes et L colonnes. L'image est divisée en éléments ou pixels définis par leurs numéros de ligne ℓ et de colonne c. Chaque entier de la matrice définit le ton de gris associé au pixel de coordonnées (ℓ, c). Ce ton varie entre zéro, qui rend l'absence de lumière et donc le noir, et une valeur maximale dite profondeur P qui rend le blanc. Les valeurs intermédiaires rendent diverses teintes de gris de plus en plus claires. On notera que le pixel de coordonnées (0, 0) est conventionnellement situé en haut et à gauche de l'image.

Remarque : L'image A est l'image test de tout le sujet.

Par exemple pour l'image élémentaire suivante  on a une échelle de 16 teintes de gris allant du noir au blanc, on a $H = 1$, $L = 16$, $P = 15$ et M ne possède qu'une seule ligne qui contient les 16 entiers de 0 à 15 en ordre croissant.

Ce TD est une adaptation à python d'un très beau sujet posé en PC 2011 (la fameuse épreuve d'informatique avant l'informatique !). Il a au moins deux objectifs : Manipuler un type d'objet créé pour le TD (introduction aux langages objet dont python fait parti), et bien entendu de manipuler les images dans un cas simple. Enfin contrairement à un écrit vous pourrez visualiser votre travail.

Usage d'une classe d'objet "image" construite pour ce TD (introduction aux langages objet)

Le fichier images.py contient la construction d'une classe d'objets **image**. Vous devez l'enregistrer dans votre répertoire contenant votre TD (ainsi que les images tes) et l'importer dans votre TD par l'appel :

```
from images import *
```

Principe d'utilisation :

L'appel de la fonction allouer(H,L,P) créé une image de taille H×L et de profondeur P. Plus précisément allouer(H, L, P) retourne une image noire de profondeur P. Cela signifie par exemple que si on tape >>> i = allouer(256,256,15)

i est alors une image de hauteur 256, largeur 256 et dont la matrice des valeurs de pixels est remplie de 0.

Remarques : J'ai volontairement créé cette fonction allouer dans la classe image pour coller au sujet original. Pour les élèves connaissant le langage objet, on accède tout aussi bien à une image par le constructeur.

On accèdera alors aux composantes d'une image *i* par les syntaxes classiques d'attribut d'un objet :

i.H renvoie la hauteur de l'image *i*

i.L renvoie la largeur de l'image *i*

i.P renvoie la largeur de l'image *i*

i.M renvoie la matrice de l'image *i*. *i.M* est pour nous un tableau numpy de taille (*H*,*L*)

On accèdera donc aux éléments de la matrice par la notation *i.M*[ℓ,*c*], sachant que les indices commencent à zéro. Un indice de ligne est donc un entier ℓ compris entre 0 et *H*−1 au sens large, tandis qu'un indice de colonne est un entier *c* compris entre 0 et *L*−1 au sens large.

On possède aussi des fonctions pour visualiser une image et lire une image :

Une fonction affiche(i) qui visualisera l'image i.

Une fonction lire_image('fichier'). Cette fonction prend n'importe qu'elle image au format png, couleur, sans transparence, et renvoie un objet image. La fonction convertit votre image **à une profondeur P = 255** de niveau de gris.

Remarques : Le sujet original (hors langage et hors de tout cours d'informatique !) proposait la phrase suivante :
[...] Enfin les candidats qui choisissent de composer dans un langage typé pourront supposer l'existence d'un type image des images [...]. C'est donc chose faite grâce au module "image.py" que je vous ai créé.

Remarques pour le travail sur machine en TD

Le travail demandé pour ce TD est double. Ce sujet étant un sujet d'écrit de concours, vous devez écrire un code propre (à l'origine sur papier) en répondant aux questions proposées.

Nous profitons bien entendu du TD et des machines à disposition pour tester les fonctions écrites et visualiser les résultats grâce à la fonction affiche(i) ajoutée bien entendu au sujet d'écrit.

Principe du répertoire courant : Pour pouvoir lire un fichier avec python, il faut bien entendu qu'il sache ou le trouver.

- Si vous utilisez IEP (c'est à dire pyzo) il faut exécuter votre session en tant que srcipt "CTRL + F5" .
- Sous IDLE (plus personne utilise IDLE ?) le répertoire courant, celui ou python va lire les fichiers, est celui de votre script. Il suffit donc de mettre votre fichier image dans le dossier du script.

Dans le dossier du TD se trouve une image test pour le tout TD :

pingouin.png **est de petite taille c'est cette image qui doit être utilisée pour tous les tests.**

Attention si vous utilisez vos images (chez vous!), elles doivent être exclusivement au format png couleur et sans transparence. Sinon la fonction élémentaire lire que j'ai fabriqué dans images.py ne fonctionnera pas. Attention aussi à leur taille qui est bien souvent bien trop grande.

Partie I opérations élémentaires

Image B Image C Image D Image E

FIG. 1: Opérations élémentaires

Question 1 Ecrire une fonction inverser(i) qui renvoie l'image inverse de i. C'est à dire que le ton d'un pixel de la nouvelle image est $i.P - v$, où v est le ton du pixel correspondant de l'image d'origine. Par exemple, l'image *B* de la figure 1 résulte de l'application de inverser à l'image A de l'introduction.

Question 2 Ecrire une fonction flipH(i) qui renvoie la transformée de l'image i par la symétrie d'axe vertical passant par le milieu de l'image. Par exemple, l'image C de la figure 1 résulte de l'application de flipH à l'image A de l'introduction.

Question 3 Ecrire une fonction flipV(i) qui renvoie la transformée de l'image i par la symétrie d'axe horizontal passant par le milieu de l'image.

Question 4 Ecrire une fonction poserV(i1, i2) qui prend en arguments deux images i1 et i2 de même largeur et profondeur, et qui renvoie la nouvelle image obtenue en posant i1 sur i2. Par exemple, l'image D de la figure 1 résulte de l'application de poserV aux images B et C.

Question 5 Ecrire une fonction poserH(i1, i2) qui prend en arguments deux images i1 et i2 de même hauteur et profondeur, et qui renvoie la nouvelle image obtenue en posant i2 à droite de i1. Par exemple, l'image E de la figure 1 résulte de l'application de poserH aux images B et C.

Partie II transferts

Certaines transformations des images sont simplement l'application d'une fonction aux tons, dont on rappelle qu'ils sont des entiers compris entre 0 et *P* (profondeur de l'image) au sens large. Une telle fonction de transfert peut s'appliquer vers des images dont la profondeur n'est pas nécessairement *P*, mais une nouvelle profondeur. Une fonction de transfert est représentée par la donnée de la profondeur cible *P2* et d'un tableau d'entiers *t* de taille $P+1$, dont les cases contiennent des entiers entre 0 et *P2* au sens large.

Question 6 Ecrire une fonction transferer(i, P2, t) qui prend en arguments une image i, ainsi qu'une fonction de transfert donnée par un entier P2 et un tableau d'entiers t. La fonction transferer renvoie une nouvelle image, de même taille que i, de nouvelle profondeur P2 et dont chaque pixel (ℓ, c) a pour ton t[i.M[ℓ,c]].

Question 7 Ecrire une nouvelle fonction inverser2(i) (Question 1) qui utilise la fonction transferer de la question précédente.

Travail sur machine en TD : Fabriquez une fonction transfert de votre choix représentée par son tableau t. Visualisez votre création.

Image A Image F Image G Image H Image I

FIG. 2: Transferts

On cherche maintenant à réduire la profondeur d'une image de P vers $P2$, avec $P2 \leqslant P$. Une technique consiste à remplacer un ton v par $v2$, tel que $v2/P2$ est le plus proche possible de v/P.
Attention : La partie entière d'un réel (float) n'est pas l'entier le plus proche de ce réel (float).

Question 8 Ecrire une fonction reduire(i,P2) qui renvoie une nouvelle image qui est i ; mais dont la profondeur est réduite à P2.
Par exemple, les images G, H et I de la figure 2 résultent des réductions à 1, 4 et 16 de la profondeur de l'image A.
Travail sur machine : Convertissez une image de votre choix en noir et blanc.

Partie III. Tramage. Conversion d une image en noir et blanc.

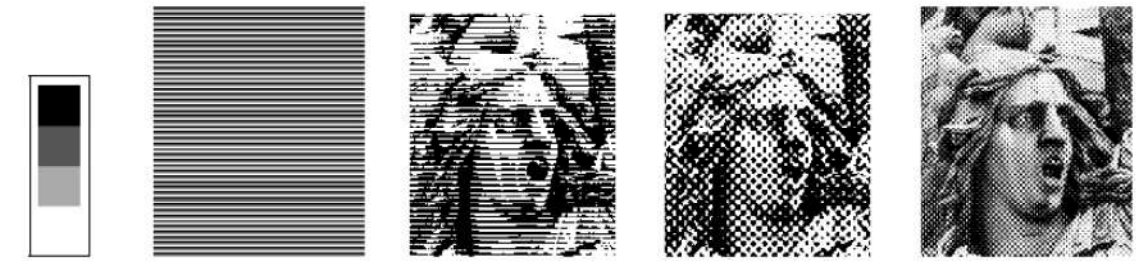

Image J Image K Image L Image M Image N

FIG. 3: Tramage

Lorsqu'il s'agit d'imprimer nos images, on se retrouve confronté à une difficulté : l'encre est noire, le papier est blanc. Il faut donc transformer les images en tons de gris en images en noir et blanc au sens strict. L'appel reduire(i,1) est un moyen de

procéder, toutefois le résultat n'est pas très satisfaisant (voir l'image G de la figure 2). Cette partie examine la technique du tramage qui produit des images en noir et blanc (c'est-à-dire de profondeur 1) plus plaisantes, telles les images L, M et N de la figure 3.

On peut voir l'image G comme produite par seuillage, les tons de gris plus grands qu'un certain seuil deviennent blancs, tandis que les autres deviennent noirs. Une idée pour améliorer le rendu des images consiste à faire varier le seuil selon les pixels. Cela revient à représenter les seuils par une matrice, en fait par une image que l'on appelle une trame. Une trame est le plus souvent constituée par la répétition d'une petite image, dite motif, répétition selon les axes de coordonnées qui finit par paver toute l'image. Par la suite, le seuillage selon une trame sera simplement désigné comme le seuillage selon le motif dont dérive cette trame. Par exemple, l'image J de la figure 3 est une échelle de gris verticale de profondeur 3 ($H = 4$, $L = 1$, $P = 3$ et la matrice M est un vecteur colonne contenant les entiers de 0 à 3 du haut vers le bas). Sa répétition produit l'image K. On notera que l'image J est présentée agrandie par rapport à l'image K. Le motif J est adéquat pour seuiller les images de profondeur 3, par exemple l'image H de la figure 2, ce qui donne l'image L de la figure 3.

Question 9 Ecrire une fonction tramer(i,m) qui prend deux images i et m en arguments, et qui renvoie l'image de profondeur 1 obtenue en seuillant l'image i selon le motif m. Le seuillage est défini précisément ainsi : étant donnés un ton v de l'image et un ton w de la trame, on obtient un pixel blanc si et seulement si $v > w$, et un pixel noir sinon.
Indication : Il ne faut pas ici construire une trame à partir du motif. Il suffit d'utiliser le modulo '%' en python qui permettra de comparer un indice de ligne ℓ (resp colonne c) de l'image i à celui du motif m modulo la hauteur de m (resp la largeur de m).

Travail sur machine en TD : Testez sur l'image de votre choix la fonction tramer(i,m) avec la trame K associée au motif J

En imprimerie on utilise rarement des trames constituées de lignes horizontales comme la trame K. On préfère les trames constituées de lignes inclinées de points, ou trames diagonales. On obtient une trame diagonale de profondeur 15 par répétition du motif 16 × 16 représenté ci-dessous, à droite :

$$\begin{pmatrix} 1 & 5 & 10 & 14 \\ 3 & 7 & 8 & 12 \\ 13 & 9 & 6 & 2 \\ 15 & 11 & 4 & 0 \end{pmatrix} =$$  $\xrightarrow{?}$ 

Le motif de droite dérive de l'image 4×4 représentée à gauche.

Question 10 Proposez le code permettant de générer le motif de droite. On utilisera pour cela les fonctions déjà créées des questions de la partie I. On appelle ici motif l'image ainsi obtenue par votre code.

Travail sur machine en TD : Vous testerez alors ce motif sur une image de votre choix. Vous pouvez utiliser la fonction lire pour convertir un fichier (**format .png**) en une image, **et ne surtout pas oublier de la réduire d'abord à la profondeur P = 15** avant de la tramer avec motif (qui est de profondeur 15) sinon vous serez très déçus ....