M4201Cin TP 1

Introduction aux images numériques

12 janvier 2018 – B. COLOMBEL

Les réponses aux questios des exercices 1 à 5 seront écrites dans l'éditeur Scinote dans un seul fichier.
Les réponses autres que des fonctions Scilab seront commentées. Il sera enregistré sous le nom :

`Nom_Prenom-TP1.sci`

et déposé dans le dossier prévu à cet effet dans AMETICE (module M3201Cin, TP1)

Le module SIVP de **Scilab** est dédié au traitement des images et des vidéos. Pour des raisons de facilité d'installation, nous allons utiliser la version 5.xx de **Scilab** disponible sous Windows.
Pour le charger :

Applications » Gestionnaire de modules - ATOMS » Traitements des images » SIVP

Cliquer sur installer puis redémarrer **Scilab**.

S'il est bien installé, le message suivant apparait :

```
Initialisation :
  Chargement de l'environnement de travail

SIVP - Scilab Image and Video Processing Toolbox
        load macros
        load gateways
        load help
        load demos
```

1 Lecture et écriture d'une image matricielle

1.1 Images en niveau de gris

Lecture et visualisation

- Pour afficher une matrice associée à une image, il suffit de taper dans la console de Scilab, l'instruction `imread('chemin␣vers␣image')`.
- Pour visualiser l'image correspondante à une matrice, on utilise l'instruction `imshow(M)` où M est la matrice des coefficients correspondants au niveau de gris de chaque pixels

Par exemple :

```
--> image0 = imread('C:\Users\colombel.b\M4201Cin\TP1\lena.pgm');
```

sous windows.

Création d'image

Réalisons une image constituée de bandes verticales allant du noir (0) au blanc (255), en passant par les 256 nuances de gris possibles.

Pour cela, on écrit une matrice constituée de ℓ colonnes de 0, ℓ colonnes de 1, ℓ colonnes de 2, etc. où ℓ représente la largeur (en pixels) d'une bande de hauteur h.

1. Écrire le programme ci-contre dans Scinotes :

```
function res = degrade(h, l)
  for i = 1:h
    for j = 1:256*l
      res(i,j) = floor((j-1)/l)
    end
  end
endfunction
```

2. Exécuter la fonction en tapant F5 puis en tapant dans la console :

```
--> Mire = degrade(60,2);
```

3. Le résultat affiché correspond-il bien au résultat attendu ?

⚠ Les nombres entiers sont écrits avec un point, ce qui signifie qu'ils sont reconnus au format « double » et écrits ainsi le logiciel ne reconnaît pas qu'il s'agit d'une matrice d'image [1]. Pour que Scilab reconnaisse que la matrice est une matrice d'image, nous allons transformer les nombres en éléments de $\mathbb{Z}/256\mathbb{Z}$ en utilisant la fonction `uint8()`.

```
--> Mire = uint8(Mire);
```

4. (a) Pour visualiser l'image associée à cette matrice, on utilise la fonction imshow, en mettant en argument le nom de la matrice.

```
--> imshow(Mire)
```

   (b) La fonction `imshow(M)` permet de voir l'image mais ne l'enregistre pas ; pour la sauvegarder, on utilise la fonction `imwrite(M, 'nom.ext')` : comme premier argument, le nom de la matrice, et comme deuxième argument, entre apostrophes, le nom que l'on souhaite donner à l'image, avec tout le chemin, sans oublier l'extension.

1.2 Image en couleurs

Lorque l'on prend une photo avec un appareil numérique, ayant par exemple deux millions de pixels, cela signifie qu'il comporte deux millions de capteurs ; chaque capteur mesure les quantités de lumière rouge, de lumière verte et de lumière bleue reçues, et les enregistre sous forme de nombres : ainsi à chaque capteur est associé un triplet (R,V,B) de nombres entiers compris entre 0 et 255, et à une image en couleur sont associées trois matrices (la première pour le rouge, la deuxième pour le vert et la dernière pour le bleu). Chaque matrice comporte, dans ce cas, deux millions de termes. Nous appellerons hypermatrice cet ensemble de trois matrices.

Les fonctions `imread`, `imshow` et `imwrite` s'utilisent avec les images en couleur comme avec les images en niveaux de gris.

1. Ce n'est pas tout à fait vrai comme nous le verrons dans la deuxième partie.

Lecture de l'hypermatrice d'une image

1. Charger l'image `test-couleurs.png` avec l'instruction `imread`. L'appeler I_1.

On obtient 3 matrices, notées respectivement (:,:,1), (:,:,2) et (:,:,3) dans la console.

2. Taper les instructions :

```
--> R = I1(:, :, 1); V = I1(:, :, 2); B = I1(:, :, 3)
--> imshow(R)
--> imshow(V)
--> imshow(B)
```

   À quoi correspondent ces trois matrices ?

Écriture d'une hypermatrice

Exercice 1 : Écrire une fonction `res = damier(m, n)` réalisant un damier alternant des pixels rouges, ayant pour coordonnées RVB : (255,0,0) et des pixels verts, ayant pour coordonnées RVB : (0,255,0). La matrice du rouge est donc une alternance de 255 et de 0, celle du vert de 0 et de 255, celle du bleu ne contient que des 0.
Les paramètres m et n représentent respectivement le nombre de lignes et le nombre de colonnes du damier.

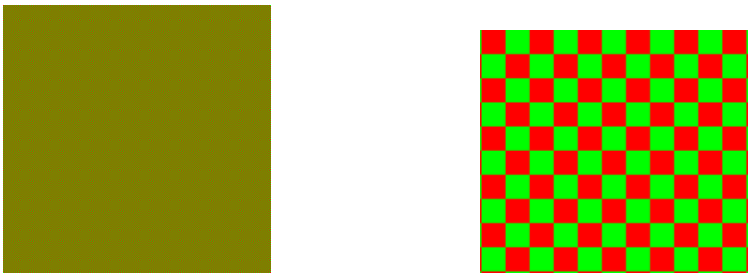


FIGURE 1 – Le damier et un zoom

Exercice 2 : L'Union Internationale des Télécommunications préconise (recommandation 601), pour les images visualisées sur un écran, de calculer le niveau de gris G (luminance) d'un pixel donné à partir de ses composantes RVB (Rouge Vert Bleu) selon la formule :

$$G = 0{,}2126R + 0{,}7152G + 0{,}0722B \qquad (G \text{ étant arrondi à l'entier le plus proche})$$

(pour un œil humain le vert parait plus lumineux que le rouge, lui-même plus lumineux que le bleu).

Écrire une fonction Scilab `Mnb = rvb2nb(M)` recevant une image couleur `(M)` comme argument et créant une nouvelle image Mnb en nuances de gris correspondant à `M`.

2 Transformations

2.1 Remarques sur les formats

Par défaut, Scilab écrit les nombres au format « `double` »(sur 64 bits). Pour écrire les éléments des matrices d'images qui sont des nombres entiers compris entre 0 et 255, c'est-à-dire les représentants compris entre 0 et 255 des éléments de $\mathbb{Z}/256\mathbb{Z}$, on a utilisé jusqu'à présent le format « `uint8` » (sur 8 bits).

En réalité, la matrice est également reconnue par Scilab comme une matrice d'images si ses éléments sont des nombres compris entre 0 et 1, écrits au format « double ». Nous avons fait ce choix du format « `uint8` » d'une part parce que lorsque l'on lit (avec la fonction `imread`) la matrice d'une image, c'est dans le format « `uint8` » qu'elle

est affichée ; et d'autre part parce que cela nous paraît préférable pour comprendre l'aspect discret du codage des couleurs.

Des erreurs d'arithmétique peuvent apparaitre si on n'y prend pas garde :

```
--> 0.7*8
 ans  =
   5.6

--> uint8(0.7)*8
 ans  =
  0

--> uint8(0.7*8)
 ans  =
  5
```

Pour éviter des erreurs d'arithmétique, il faudra donc être attentif au type de nombre sur lesquels on travaille, et pour calculer `k*M`, on commencera par convertir la matrice en une matrice de nombres au format `double` avec la fonction `double()` et on reviendra ensuite à une matrice d'image avec la fonction `uint8()`. Par exemple :

```
--> M = imread('C:\Users\colombel.b\M4201Cin\TP1\lena.pgm');
--> res = 0.7*double(M);
--> imshow(uint8(res));
```

2.2 Symétries

Exercice 3 : Écrire trois fonctions en **Scilab** qui permettent de passer de l'image initiale `dollar.pgm` aux images transformées :

Image originale

sysmétrie horizontale

symétrie verticale

transposition

Remarque. Si M est une matrice de dimension $n \times p$, alors la commande `M(i,:)` renvoie la i^e ligne de la matrice M et la commande `M(:,j)` renvoie la j^e ligne de `M` ce qui permet de travailler direcetment avec les lignes et les colonnes.

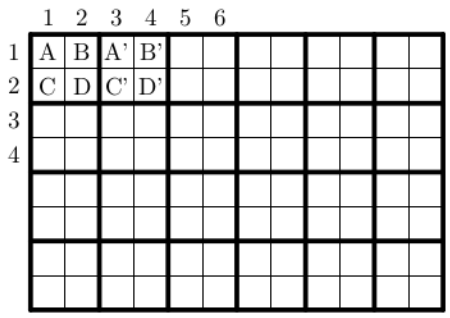
```
--> M = rand(5,6)
 M  =

   0.2113249   0.6283918   0.5608486   0.2320748   0.3076091   0.2922267
   0.7560439   0.8497452   0.6623569   0.2312237   0.9329616   0.5664249
   0.0002211   0.685731    0.7263507   0.2164633   0.2146008   0.4826472
   0.3303271   0.8782165   0.1985144   0.8833888   0.312642    0.3321719
   0.6653811   0.068374    0.5442573   0.6525135   0.3616361   0.5935095
```

```
--> M(2,:)
 ans  =

   0.7560439   0.8497452   0.6623569   0.2312237   0.9329616   0.5664249


--> M(:,4)
 ans  =

   0.2320748
   0.2312237
   0.2164633
   0.8833888
   0.6525135
```

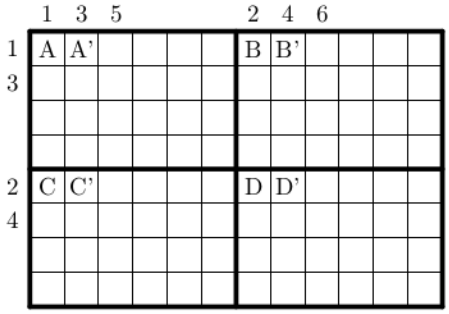
2.3 Transformation du photomaton

Exercice 4 : **Principe** :

- on travaille sur des matrices ayant un nombre pair de lignes n et un nombre pair de colonnes m ;
- on découpe la matrice initiale en petits carrés de 2×2 ;
- on découpe la matrice d'arrivée en quatre matrices ayant $\frac{n}{2}$ et lignes et $\frac{m}{2}$ colonnes ;
- on range les 4 éléments de chaque petit carré comme sur la figure ci dessous :



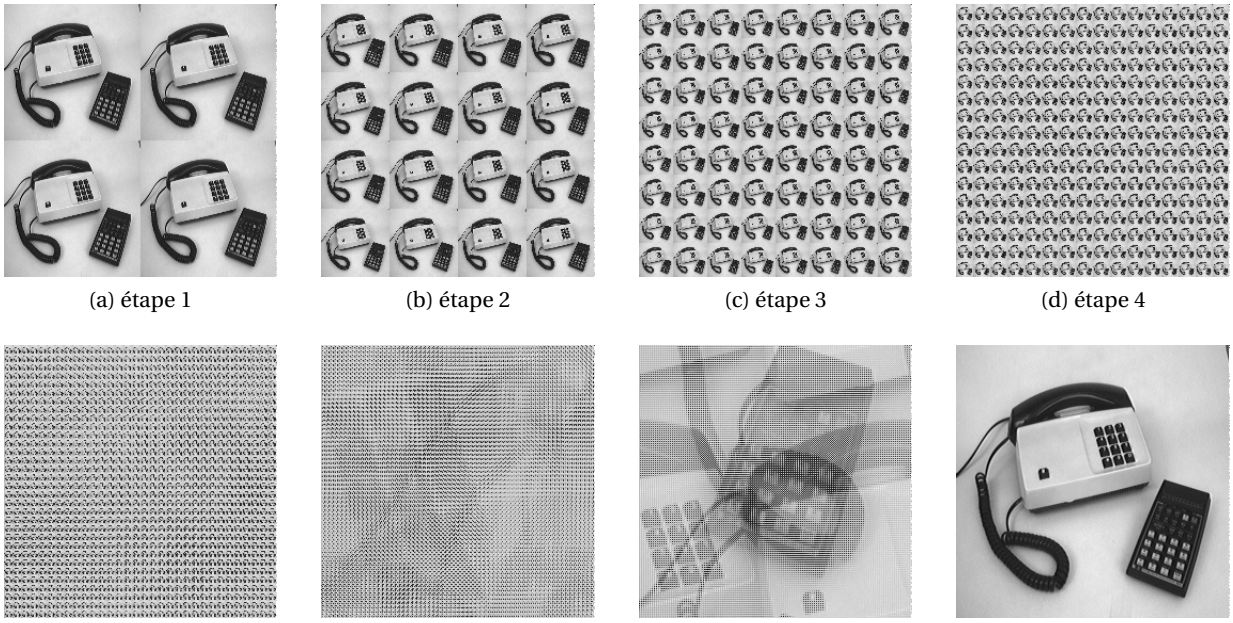
Matrice initiale



Matrice photomaton

Les lignes impaires de la matrice initiale se retrouvent donc dans la moitié supérieure de la matrice photomaton, les lignes paires dans la moitié inférieure. De même les colonnes impaires se retrouvent dans la partie gauche, et les colonnes paires dans la partie droite.

1. Écrire une fonction `res = photomaton(M)` qui, à une image `M` associe son image transformée.
2. (a) Itérer plusieurs fois le procédé avec l'image `phone.pgm`.
   Que remarque-t-on ?
   (b) Tester avec d'autres images. Que peut-on conjecturer ?
3. Écrire une fonction `res = diaporama(M)` qui calcule le nombre d'itérations nécessaires pour revenir à l'image de départ.
4. (Bonus) Démontrer la conjecture.



(a) étape 1 (b) étape 2 (c) étape 3 (d) étape 4

(e) étape 5 (f) étape 6 (g) étape 7 (h) étape 8

FIGURE 2 – Transformation du photomaton