Examen LS4 2009/2010 - session 1

28 mai 2010

---

durée : 2h45
Les documents sont interdits. Vous devez éteindre et ranger vos téléphones.

Les programmes sont à faire en PYTHON .
Vous êtes priés de respecter les dénominations imposées dans l'énoncé. Vous pouvez à tout moment introduire une fonction annexe si vous en ressentez le besoin. A tout moment vous pouvez supposée écrite une fonction *précédemment* demandée dans l'énoncé.

Les figures illustrant l'énoncé se trouvent toutes en dernière page.

Le but de cet énoncé est d'écrire une librairie de gestion d'images et de vidéos.

Une *image* est vue comme une matrice rectangulaire de pixels (une liste de listes), chaque pixel étant représenté par l'intensité des couleurs rouge, vert et bleu (un tuple de trois entiers qu'on supposera compris entre 0 et 255), un exemple est donné en figure 1. Une *vidéo* correspond à une liste d'images de mêmes dimensions.

Pour mémoire, le blanc est représenté par $(255, 255, 255)$ et le noir par $(0, 0, 0)$. Les diverses nuances de gris correspondent à des triplets d'entiers identiques. Le rouge le plus clair est obtenu grâce à $(255, 0, 0)$.

1 Travail sur les images

Exercice 1 – *Extraction*

On suppose qu'on a un ensemble d'images (sous la forme d'une séquence PYTHON) et qu'on veut en extraire un sous-ensemble qui correspond à certains critères.

1. Ecrire une fonction `est_verticale` qui prend en argument une image et renvoie vrai si la hauteur de l'image est supérieure à sa largeur (et faux sinon).
2. Ecrire une fonction `est_noir_et_blanc` qui prend en argument une image et renvoie vrai si l'image donnée en argument est en noir et blanc[1] (et faux sinon).
3. On dit qu'une image est *rouge* si pour au moins $2/3$ de ses pixels le coefficient du rouge est au moins le double des coefficients du vert et du bleu. Ecrire une fonction `est_rouge` qui prend une image en argument et détermine si celle-ci est rouge.
4. Ecrire une fonction `extraire` qui prend en argument un ensemble d'images (sous la forme d'une séquence PYTHON) et un nom de fonction représentant une propriété sur les images, et renvoie la liste des images pour lesquelles cette propriété est vraie.

[1] Au sens photographique du terme : en niveaux de gris.

Exercice 2 – *Réarrangement*

On suppose qu'on a un ensemble d'images que l'on veut ordonner de la plus claire à la plus foncée.

On appelle *intensité moyenne* d'une image la moyenne de tous les coefficients de couleurs qui apparaissent dans ses pixels (rouge, vert et bleu de façon indifférente).

Un image sera considérée comme *plus claire* qu'une autre si son intensité moyenne est supérieure à l'intensité moyenne de l'autre. On dira dans ce cas que la deuxième image est *plus foncée* que la première.

1. Ecrire une fonction `intensite_moyenne` qui renvoie l'intensité moyenne de l'image fournie en argument.
2. Ecrire une fonction `est_plus_claire` qui prend en argument deux images et détermine si la première est plus claire que la deuxième.
3. Ecrire une fonction `degrade` qui prend en argument une liste d'images `L` et renvoie la liste ordonnée des images de `L`, de la plus claire à la plus foncée. Pour cela, utilisez l'algorithme récursif suivant :
   - si la liste ne contient qu'une image, on la renvoie telle quelle,
   - sinon
     - la première image est appelée *pivot*,
     - on partitionne la liste `L` en `L1`, les images plus claires que le pivot, et `L2`, les images plus foncées que le pivot,
     - `L1` et `L2` sont ordonnées récursivement,
     - on remet tout ensemble pour obtenir la solution.

Exercice 3 – *Transformation*

Pour toutes les transformations, on demande la création d'une nouvelle image.

réduction d'une image On veut réduire la taille d'une image en divisant sa hauteur ou sa largeur par un entier.

1. Ecrire de même une fonction `reduction_horizontale` qui permette de diviser la largeur de l'image par un entier. Un pixel de l'image d'arrivée sera obtenu en moyennant les pixels correspondants de l'image de départ, comme montré en figure 2.
2. Ecrire une fonction `reduction_verticale` qui permette de diviser la hauteur de l'image par un entier.

agrandissement d'une image On veut agrandir la taille d'une image en multipliant sa hauteur ou sa largeur par un entier. L'agrandissement sera fait naïvement en répétant plusieurs fois le pixel de départ.

3. Ecrire de même une fonction `agrandissement_horizontal` qui permette de multiplier la largeur de l'image par un entier. Les coefficients des pixels de l'image d'arrivée seront obtenus en recopiant les coefficients du pixel correspondant de l'image de départ, comme montré en figure 3.
4. Ecrire une fonction `agrandissement_vertical` qui permette de multiplier la hauteur de l'image par un entier.

5. Le résultat obtenu par les deux fonctions précédentes n'est pas très joli car le grain de l'image est agrandi. Pour éviter ce genre d'effet, on peut remplacer la valeur en un pixel par une moyenne associant les valeur des pixels autour de celui-ci, comme illustré en figure 4
   Ecrire une fonction `anti_aliasing` qui implémente cette transformation.

regarder par le trou d'une serrure On suppose qu'on dispose d'une image `serrure` qui représente la forme d'un trou de serrure. On veut regarder une autre image à travers ce trou de serrure, comme montré en figure 5. Si la serrure n'a pas la même taille que l'autre image, on commencera par déformer la serrure à la taille de l'autre image. On pourra pour cela supposer que les rapports de longueurs entre les deux images est entier.

6. Ecrire une fonction `par_la_serrure`.

2 Travail sur des vidéos

On suppose l'existence des fonctions suivantes :
- `gif2images` qui prend en argument le nom d'un fichier contenant un gif animé et renvoie une liste d'images,
- `images2gif` qui prend en argument un nom de fichier et une liste d'images et crée le gif animé.

Exercice 4 – *Suppression des écrans publicitaires*

On dispose d'un enregistrement télévisé, sous forme d'un gif animé, dans lequel on veut supprimer les écrans publicitaires.

Un écran publicitaire est entouré d'écrans monochromes, c'est-à-dire des images quasiment d'une seule couleur. Par convention on considèrera qu'une image est monochrome si 99% de sa surface est d'une seule et même couleur.

Ecrire une fonction `stop_la_pub` qui prend en argument un nom de fichier contenant un gif animé et crée un nouveau fichier contenant le même film dont on a supprimé les écrans publicitaires.

(On pourra supposer que le document ne commence pas par une publicité.)

Exercice 5 – *Segmentation en plans*

On cherche à segmenter une vidéo en plans en repérant automatiquement les points de montage. Pour cela, on suppose que le changement entre deux plans est brusque.

On définit la distance entre deux images par le nombre de pixels différents entre ces images.

1. Ecrire une fonction `distance_moyenne` qui calcule la distance moyenne entre deux images consécutives d'une vidéo.
2. On considère qu'il y a changement de plan si la distance entre deux images consécutives est au moins 10 fois supérieure à la distance moyenne. Ecrire une fonction `segmentation` qui segmente une vidéo en plans. Vous créerez une vidéo par plan. Cette fonction prendra en argument le nom d'un fichier contenant un gif animé, créera un fichier contenant un gif animé par plan et retournera le nombre de plans trouvés.

3 Les exemples

```
[[(0,0,0),(255,255,255),(255,0,0)],
 [(0,0,255),(0,255,0),(255,0,0)]]
```

$(0,0,0)$	$(255,255,255)$	$(255,0,0)$
$(0,0,255)$	$(0,255,0)$	$(255,0,0)$

$\longrightarrow$ 

FIG. 1 – exemple d'image

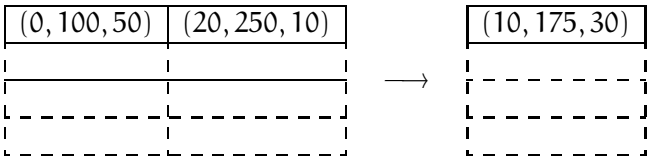

FIG. 2 – réduction horizontale (coefficient :2)

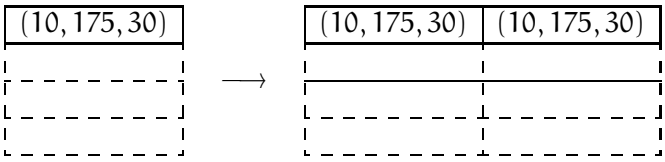

FIG. 3 – agrandissement horizontal (coefficient :2)

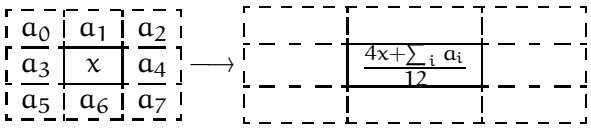

FIG. 4 – anti-aliasing

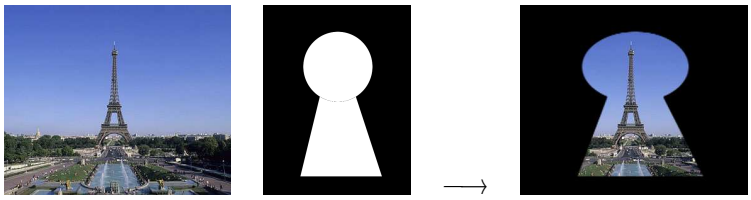

FIG. 5 – regarder par le trou d'une serrure