# Examen LS4 2009/2010 - session 2

21 juin 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** – *Couleurs*

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

**passage au noir et blanc** On veut transformer une image a priori en couleurs en image en noir et blanc[1]. Pour cela, chaque pixel de la transformée sera gris, la valeur de ses coefficients étant déterminée par par sa luminance[2] qui est donnée par la formule

$$0,299\mathrm{r} + 0,587v + 0,114\mathrm{b},$$

où $\mathrm{r}$ (resp. $v$ et $\mathrm{b}$) représente le coefficient du pixel d'origine associé au rouge (resp. vert et bleu).

1. Ecrire une fonction `noir_et_blanc` qui prend en argument une image et renvoie une image représentant la même illustration en noir et blanc.

[1] Au sens photographique du terme, c'est-à-dire en niveaux de gris.
[2] C'est la composante d'une couleur correspondant à la luminosité.

**anti yeux rouges** Il s'agit ici de détecter automatiquement des yeux rouges sur une image et de les remplir en noir, comme montré sur la figure 4.

On appelle *zone colorée* d'une image une partie connexe (c'est-à-dire en un seul morceau) où tous les pixels ont même couleur et qui est maximale pour cette propriété (aucun pixel de cette même couleur ne touche cette région), voir figure 2 pour un exemple. On veut pouvoir modifier la couleur d'une zone colorée.

2. Ecrire une fonction `remplissage` qui prend en argument une image, les coordonnées d'un pixel $p$ et une couleur $c$ et renvoie une nouvelle image dans laquelle la zone colorée contenant le pixel $p$ a été repeinte avec la couleur $c$.

On appelle *diamètre* d'une zone colorée la plus grande distance[3] entre deux points de cette zone.

3. Ecrire une fonction `diametre` qui prend en argument une image et les coordonnées d'un pixel $p$ et renvoie le diamètre de la zone colorée contenant $p$.

On appelle *distance* entre deux zones colorées $z_1$ et $z_2$ la plus petite distance entre un point de $z_1$ et un point de $z_2$.

4. Ecrire une fonction `distance` qui prend en argument une image et les coordonnées de deux pixels $p_1$ et $p_2$ et renvoie la distance entre la zone colorée contenant $p_1$ et celle contenant $p_2$.

Une paire de zones colorées est considérée comme une *paire d'yeux rouges* si elles sont toutes deux rouges (c'est-à-dire que la couleur de tous leurs pixels est $(255, 0, 0)$) que leurs diamètres ne diffèrent pas de plus de 10% et que leur distance est comprise entre 5 et 10 fois la moyenne de leurs diamètres.

5. Ecrire une fonction `paire_yeux_rouges` qui prend en argument une image et les coordonnées de deux pixels $p_1$ et $p_2$ et détermine si le couple de zones colorées contenant $p_1$ et $p_2$ correspond à des yeux rouges.
6. Ecrire une fonction `suppression_yeux_rouges` qui prend en argument une image et renvoie une nouvelle image dans laquelle les yeux rouges ont été supprimés (attention, ils peut y avoir plusieurs paires d'yeux rouges).

**Exercice 2** – *Transformation du photomaton*

Le principe de la *tranformation du photomaton* est de partir d'une image initiale et d'obtenir une image recomposée en 4 images rétrécies récursivement.

Il ne s'agit pas de réduire la taille de l'image de départ, mais de déplacer des pixels, comme montré en figure 3 : l'image de départ est découpée en carrés de $2 \times 2$ pixels, le pixel en haut à gauche (resp. en haut à droite / en bas à gauche / en bas à droite) sert à recomposer une image de taille moitié située en haut à gauche (resp. en haut à droite / en bas à gauche / en bas à droite).

Si on applique suffisament de fois cette transformation, on retombe sur l'image de départ, comme illustré en figure 5.

Ecrire une fonction qui prend en argument une image dont on supposera que les longueurs des côtés sont paires et qui renvoie sa transformée par la transformation du photomaton.

[3] Pour rappel, la formule donnant la distance entre deux points $M(x, y)$ et $M'(x', y')$ est $\sqrt{(x - x')^2 + (y - y')^2}$.

## 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 3** – *Accélération*

On dispose d'un enregistrement vidéo, sous forme d'un gif animé, que l'on veut accélérer. La stratégie à utiliser est de garder des images régulièrement espacées.

Ecrire une fonction `acceleration` qui prend en argument le nom d'un fichier contenant un gif animé et un entier correspondant au coefficient de réduction (par exemple si la vidéo de départ dure 20 minutes et le coefficient est 5, alors la vidéo d'arrivée doit durer 4 minutes) et crée un nouveau fichier contenant un gif animé de la vidéo accélérée.

**Exercice 4** – *Images subliminales*

Au cinéma, il y a 24 images par seconde. En insérant une image hors contexte (promotionnelle, par exemple), cette dernière ne s'affichera que $0,04$ seconde et ne pourra donc pas être perçue consciemment par le spectateur mais pourrait être enregistrée par le cerveau malgré tout[4]. Une telle image insérée hors contexte est appelée *image subliminale*.

Le but de cet exercice est d'extraire d'un film donné sous la forme d'un fichier contenant un gif animé les probables images subliminales. Pour cela il faut être capable de repérer les images subliminales. Une image subliminale étant hors contexte, on peut supposer qu'elle est très différente des images qui l'entourent qui, elles, sont censées être proches.

On considère la suite des valeurs successives des coefficients $(r, v, b)$ associés à un pixel donné de l'image. L'*écart-type* $(\sigma_r, \sigma_v, \sigma_b)$ de cette suite est donné par la formule

$$\sigma_r^2 = E\big((r - E(r))^2\big)$$

(et de façon équivalente pour $\sigma_v$ et $\sigma_b$), où $E(X)$ désigne la moyenne (arithmétique) de la valeur $X$.

1. Ecrire une fonction `moyenne` qui calcule la moyenne d'une séquence d'entiers donnée en argument.
2. Ecrire une fonction `ecart_type_carre` qui calcule le carré de l'écart type d'une séquence d'entiers donnée en argument.
3. On considère qu'une image est hors contexte (donc est une image subliminale) si pour au moins 60% de ses pixels la différence, sur au moins une couleur, avec chacune de ses deux voisines est supérieur à 5 fois l'écart-type associé à ce pixel dans la suite d'images. Ecrire une fonction `extraire_images_subliminales` qui prend en argument un nom de fichier contenant un gif animé et renvoie une liste d'images correspondant aux images subliminales contenues dans l'animation.

[4] extrait de `http://fr.wikipedia.org/wiki/Message_subliminal`

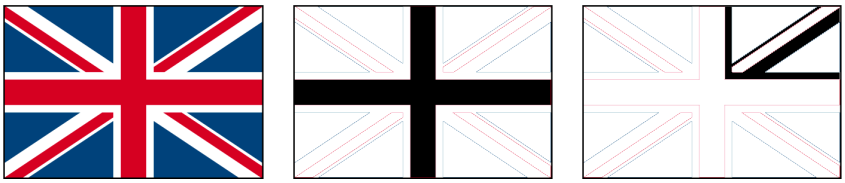
# 3 Les exemples

```
[[(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



(a) image de départ (b) zone colorée (c) autre zone colorée

FIG. 2 – exemples de zones colorées d'une image

| | | | |
|---|---|---|---|
| 1a | 1b | 2a | 2b |
| 1c | 1d | 2c | 2d |
| 3a | 3b | 4a | 4b |
| 3c | 3d | 4c | 4d |
| 5a | 5b | 6a | 6b |
| 5c | 5d | 6c | 6d |
| 7a | 7b | 8a | 8b |
| 7c | 7d | 8c | 8d |

$\longrightarrow$

| | | | |
|---|---|---|---|
| 1a | 2a | 1b | 2b |
| 3a | 4a | 3b | 4b |
| 5a | 6a | 5b | 6b |
| 7a | 8a | 7b | 8b |
| 1c | 2c | 1d | 2d |
| 3c | 4c | 3d | 4d |
| 5c | 6c | 5d | 6d |
| 7c | 8c | 7d | 8d |

FIG. 3 – principe de la transormation du photomaton

(a) photo avec yeux rouges (b) photo sans yeux rouges

FIG. 4 – suppression des yeux rouges

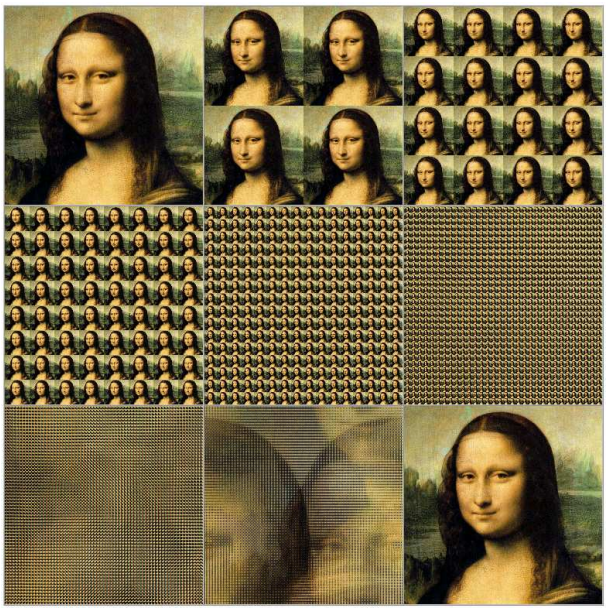


FIG. 5 – exemple d'applications successives de la transformée du photomaton (source : `http://upload.wikimedia.org/wikipedia/commons/1/17/Photomaton.png`)