

Devoir maison Facultatif n°1 - A rendre le jeudi 6 novembre 2025

Problème : Autofocus sur les appareils photos numériques

Nous nous intéressons dans ce problème à l'étude de différentes méthodes permettant de réaliser l'autofocus sur les appareils photos numériques. L'autofocus consiste à régler de manière automatique la netteté de l'image avant d'effectuer la prise de vue.

Pour rédiger les scripts, on supposera que tous les modules nécessaires ont été importés. Par exemple, toutes les fonctions de `numpy` ou celles de `matplotlib.pyplot`. Les fonctions importées peuvent être utilisées sans préfixe (`plot`, `arange`...).

Pour réaliser l'autofocus, il existe deux méthodes principales : la mesure du contraste, et la mesure de détection de phase.

Mesure de contraste.

Le principe de cette méthode repose sur le fait qu'une image bien mise au point présente un maximum de contraste. Plus précisément l'écart de valeur entre les pixels est la plus grande.

En fonctionnement, l'appareil mesure le contraste, puis par tâtonnement, va déplacer l'objectif jusqu'à détecter le maximum.

Représentation d'une image.

Une image matricielle est représentée sous la forme d'une matrice. Chaque élément de la matrice correspondra à un pixel qui est généralement représenté par une liste de 3 entiers naturels, représentant les composantes rouge, verte et bleue ; c'est le codage RVB. Chaque valeur est représentée par un entier allant de 0 à 255. On suppose, dans cette partie, travailler avec un appareil photo doté d'un capteur de 48 MPixels, c'est-à-dire que la photo sera composée de 48 millions de pixels. On suppose que le stockage des photos se fait en mode RAW sans compression, c'est-à-dire que l'on stocke directement ce que récupère le capteur.

L'image sera représentée en mémoire dans une variable `I`, de type liste de liste de liste Python classique ou de type `numpy.ndarray`. Dans les deux cas, on accédera aux données RVB d'un pixel de coordonnées (i, j) , par l'instruction `I[i][j]`. Le candidat choisira librement le type qu'il préfère manipuler.

Question 1 : Préciser l'espace mémoire nécessaire pour stocker la valeur d'une composante, puis celle d'un pixel et enfin celle d'une image en Mo (= 1000 ko) ou Mio (= 1024 ko).

Conversion en niveau de gris.

A partir de maintenant, nous supposons que l'intensité de chaque couleur est codée par un flottant entre 0 et 1 (qui correspond au nombre entier entre 0 et 255, divisé par 255).

La première étape de la détection du contraste est de convertir l'image en niveau de gris pour n'obtenir qu'une valeur par pixel.

Cette transformation s'opère en plusieurs étapes :

- chaque composante R, V et B, qui n'est pas linéaire en terme d'intensité lumineuse sur le rendu, est d'abord transformée dans un espace linéaire. Une composante C sera transformée selon la définition suivante :
 - $C_{lin} = \frac{C}{12,92}$ si $C \leq 0,04045$;
 - $C_{lin} = \left(\frac{C + 0,055}{1,055} \right)^{2,4}$ sinon.
- la valeur du niveau de gris Y_{lin} en échelle linéaire sera calculée à partir des valeurs linéarisées par :
$$Y_{lin} = 0,2126R_{lin} + 0,7152V_{lin} + 0,0722B_{lin}$$
- on repasse dans l'espace non linéaire avec le calcul suivant :
 - $Y = 12,92Y_{lin}$ si $Y_{lin} \leq 0,0031308$;
 - $Y = 1,055Y_{lin}^{1/2,4} - 0,055$ sinon.

Question 2 : Ecrire une fonction `Clinear(val)` qui prend en argument une valeur de l'espace non linéaire et qui renvoie la valeur linéarisée.

Question 3 : Ecrire une fonction `Y(pix)` qui prend en argument une liste de trois valeurs correspondant à un pixel au format RVB et qui renvoie la valeur Y du niveau de gris dans l'espace non linéaire.

Question 4 : Ecrire une fonction `NiveauxGris(I)` prenant en argument une image I au format RVB et qui renvoie une image de même hauteur et de même largeur que I , dont chaque élément est une valeur correspondant au niveau de gris d'un pixel.

Pour détecter le contraste d'une image en niveau de gris, on va comparer pour chaque pixel les valeurs autour de celui-ci. Plus l'écart est grand (ce qui est le cas quand l'image est nette), plus les pixels autour du pixel de référence ont une valeur différente ; on calcule d'une certaine manière la dérivée en chaque pixel.

Cette opération est réalisée par un filtre de Sobel. Partant d'une image I , on extrait les pixels autour du pixel (i, j) sous

la forme d'une matrice 3×3 , notée $I_e = \begin{pmatrix} I_{i-1,j-1} & I_{i-1,j} & I_{i-1,j+1} \\ I_{i,j-1} & I_{i,j} & I_{i,j+1} \\ I_{i+1,j-1} & I_{i+1,j} & I_{i+1,j+1} \end{pmatrix}$.

On réalise ensuite une convolution entre cette matrice et la matrice de filtration.

On définit la convolution entre deux matrices A et B de taille 3×3 par $A \circledast B = \sum_{i=0}^2 \sum_{j=0}^2 A_{ij} B_{ij}$.

On réalise une filtration selon les deux directions de l'image.

On donne les deux matrices de filtration : $G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ et $G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$.

Enfin on calcule une norme euclidienne des deux convolutions dont on prend la partie entière pour obtenir la valeur du contraste c du pixel (i, j) : $c = \text{int} \left(\sqrt{(I_e \circledast G_x)^2 + (I_e \circledast G_y)^2} \right)$.

Pour terminer, on calcule le contraste de cette manière pour chaque pixel intérieur à l'image (on ne calcule rien sur les bords pour simplifier), puis on réalise la moyenne des valeurs pour obtenir l'indice de contraste de référence c_{ref} .

Question 5 : Ecrire une fonction `convolution(A, B)` prenant en argument deux matrices de taille 3×3 et qui renvoie la valeur du produit de convolution.

Question 6 : Ecrire une fonction `contraste_pixel(I, i, j)` prenant en argument une image I au format niveaux de gris et les coordonnées du pixel (i, j) qui renvoie la valeur du contraste défini précédemment par la quantité c .

Question 7 : Ecrire une fonction `contraste(I)` prenant en argument une image `I` au format niveau de gris et qui renvoie la valeur du contraste de référence c_{ref} .

Détection de phase.

La méthode par détection de phase consiste à mesurer une petite partie de l'image par deux capteurs différents. Quand les deux mesures sont identiques, l'image est nette. En calculant la différence de phase entre les deux mesures, il est possible de calculer directement la valeur dont doit se déplacer l'objectif.

On suppose que l'on dispose de deux capteurs de longueurs 100 pixels (dont les valeurs seront stockées dans des listes) et que l'on a exactement la même séquence de valeurs mais décalées de quelques pixels. La différence de phase est le nombre de pixels (donc le nombre d'indices de décalage entre les deux listes) permettant de retrouver les mêmes séquences de pixels.

L'objectif est donc de comparer deux listes de valeurs, de même dimension, extraites des listes de données des deux capteurs. Trois cas peuvent se poser en fonction de la valeur du décalage noté `dec` et sont illustrés sur la figure ci-dessous. Les sous-listes extraites sont grisées.

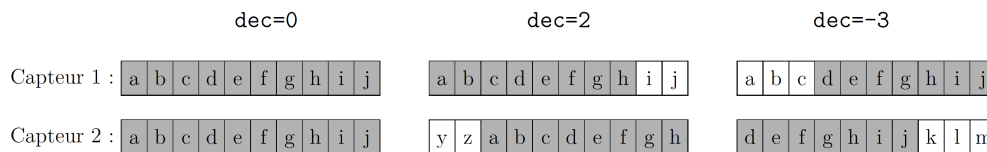


FIGURE 1 – Illustration du décalage (sur l'alphabet)

Question 8 : Ecrire une fonction `extraction(L1, L2, dec)` prenant en argument deux listes `L1` et `L2` ainsi qu'une valeur entière de décalage et qui renvoie deux sous-listes à comparer de longueur `len(L1)-abs(dec)`, conformément aux règles présentées ci-dessus. On supposera que les deux listes `L1` et `L2` sont de même taille.

Question 9 : Ecrire une fonction `comparaison(L1, L2)` (n'utilisant pas la comparaison interne de Python entre listes) prenant en argument deux listes `L1` et `L2` qui renvoie `True` si les listes sont identiques et `False` sinon.

On suppose que le décalage est compris entre `-80` et `80`.

Question 10 : Ecrire une fonction `recherche_decalage(L1, L2)` prenant en argument deux listes `L1` et `L2` et qui renvoie la valeur du décalage ou `None` s'il n'existe pas.

Question 11 : Evaluer la complexité de la fonction `recherche_decalage(L1, L2)` en prenant en compte le nombre de comparaisons en fonction de `n` la taille des listes et de `m` le nombre de décalage maximal à prendre en compte (161 dans notre exemple) dans le meilleur et dans le pire des cas.

En pratique, il n'y a pas de raisons que les deux capteurs CCD mesurent exactement les mêmes valeurs décalées. Les valeurs mesurées décalées seront approximativement les mêmes. Pour résoudre ce problème, on va calculer une erreur quadratique moyenne entre les deux sous-listes et puis chercher le minimum en fonction du décalage. Pour deux sous-listes `L1` et `L2`

de taille `n`, l'erreur sera définie par :
$$erreur = \frac{1}{n} \sum_{i=0}^{n-1} (L1_i - L2_i)^2$$

Question 12 : Ecrire une fonction `erreur(L1, L2)` qui retourne l'erreur quadratique définie ci-dessus.

Question 13 : Ecrire une fonction `recherche_decalage_2(L1, L2)` qui cherche le minimum de l'erreur pour des décalages de `-80` à `80` et qui retourne la valeur de ce décalage.