Master Automatique, Robotique, Master 1 – 1er semestre
UE de Traitement des images (3 ECTS) - UM4RBI11-Image1

Responsable de l'UE
Prof. Daniel Racoceanu
Sorbonne Université, Faculté des Sciences et Ingénierie

NB : L'implémentation des solutions se fera en Python, de préférence dans un environnement de type Google Collab ou Jupyter (afin de ne pas gérer de soucis d'installation sur vos propres machines). Les commentaires sont à introduire sous forme de commentaire dans le code qui sera ainsi soumis dans des dossiers spécifiques (au TP et au groupe), en ligne, sur Moodle. En règle générale, pour cette première séance, je vous conseille de travailler les valeurs numériques des pixels au niveau bas, avant d'éventuellement appliquer des fonctions Python de haut niveau. The implementation of the solutions will be carried out in Python, preferably in a Google Colab or Jupyter environment (so as to avoid dealing with installation issues on your own machines). Comments must be introduced in the form of annotations within the code, which will then be submitted in specific folders (corresponding to the lab session and the group), online, via Moodle.As a general rule, for this first session, I recommend working with the numerical values of the pixels at a low level before eventually applying higher-level Python functions.

TP 1

1. Implémenter une fonction qui extrait la **composante rouge** de l'image.

   *Implement a function able to extract the **red component** of an image.*

Extraction de la composante rouge - Red component extraction

2. Implémentez une fonction qui va créer le **négatif d'une image** (soustrayez de 255 chaque composante RGB de chaque pixel. Cela inverse l'intensité de chaque couleur : le clair devient foncé et vice versa).

   *Implement a function able to generate the **negative of an image** (from 255, you will so need to substract the intensity's component – RGB - of each pixel. This will invert each color's intensity of each pixel: the bright will become dark and vice versa).*

Image « négative » - « Negative » image

3. **Convertir une image couleur en image en niveaux de gris**. Dans une telle image, chaque pixel est noir, blanc, ou a un niveau de gris entre les deux. Cela signifie que les trois composantes ont la même valeur. La formule standard donnant le niveau de gris en fonction des trois composantes couleur est :

$$gray = 0.299 * red + 0.587 * green + 0.114 * blue$$

Convert a color image in a gray level image. *In such an image, each pixel is black, white or has a gray level between 0 and 1. This means that the components have the same value. The standard formula giving the grey level function of the components is:*

$$gray = 0.299 * red + 0.587 * green + 0.114 * blue$$

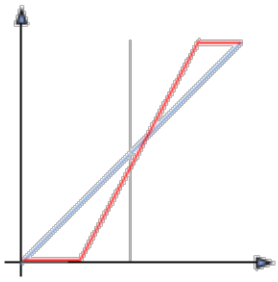
Image en niveaux de gris - Gray level image

4. **Convertir une image couleur en nuances de sépia**. En photographie, le sépia est une qualité de tirage qui ressemble au noir et blanc, mais avec des variations de brun, et non de gris. Dans la transformation d'une image couleur en une image en nuances de sépia, on tient compte des transfomations suivantes :
outputRed = (inputRed * .393) + (inputGreen *.769) + (inputBlue * .189) outputGreen = (inputRed * .349) + (inputGreen *.686) + (inputBlue * .168) outputBlue = (inputRed * .272) + (inputGreen *.534) + (inputBlue * .131)
NB : en cas de saturation (intensité résultante>255), mettre la valeur à 255.

Convert a color image to sepia tones. *In photography, sepia is a print quality that looks like black and white, but with variations of brown, not gray. In the transformation of a color image into an image in sepia tones, we consider the next transformations:*
outputRed = (inputRed * .393) + (inputGreen *.769) + (inputBlue * .189) outputGreen = (inputRed * .349) + (inputGreen *.686) + (inputBlue * .168) outputBlue = (inputRed * .272) + (inputGreen *.534) + (inputBlue * .131)
NB: in case of saturation (resulting intensity>255), set the value to 255.

Effet « sepia » - « sepia » effect

5. **Contrast**". Pour chaque composante de chaque pixel, appliquez les règles suivantes :
   a. Si la valeur est plus petite que 30, assignez la valeur 0.
   b. Si la valeur est plus grande que 225, assignez la valeur 255.
   c. Les valeurs *c* comprises entre 30 et 225 seront recalculées avec la formule : (255.0 / 195.0) * (*c* - 30) + 0.5



Avec une calculatrice, vous constaterez que, par exemple : 30 devient 0, 225 devient 255, 45 devient 20, 180 devient 196, 120 devient 118. Les valeurs "s'écartent" les unes des autres, ce qui augmente le contraste.

*"**Contrast**". For each component of each pixel, apply the following rules:*
   a. If the value is less than 30, assign the value 0.
   b. If the value is greater than 225, assign the value 255.
   *c. The c values between 30 and 225 will be recalculated with the formula: (255.0 / 195.0) * (c - 30) + 0.5.*

With a calculator, you will find that, for example: 30 becomes 0, 225 becomes 255, 45 becomes 20, 180 becomes 196, 120 becomes 118. The values "deviate" from each other, which increases the contrast.

Look Up Table LUT (courbe tonale) and "Contrast" effect

6. **Seuillage.** Le seuillage d'image est la méthode la plus simple de segmentation d'image. À partir d'une image en niveau de gris, le seuillage d'image peut être utilisé pour créer une image comportant uniquement deux valeurs, noir ou blanc (monochrome). On remplace un à un les pixels d'une image par rapport à une valeur seuil fixée (par exemple 123). Ainsi, si un pixel à une valeur supérieure au seuil (par exemple 150), il prendra la valeur 255 (blanc), et si sa valeur est inférieure (par exemple 100), il prendra la valeur 0 (noir). Appliquez aussi, une méthode de seuillage automatique / adaptatif et comparez le seuil obtenu avec le seuil suggéré. Tracez l'histogramme de l'image et analysez le seuil automatique par rapport à la configuration de l'histogramme. Faites un commentaire dans le code à ce sujet.

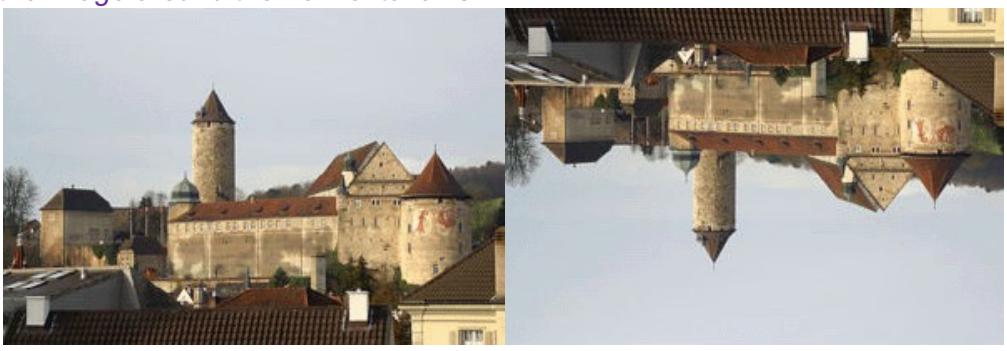
Thresholding. Image thresholding is the simplest method of image segmentation. From a grayscale image, image thresholding can be used to create an image with only two values, black or white (*monochrome*). The pixels of an image are replaced one by one with respect to a fixed threshold value (for example 123). Thus, if a pixel has a value greater than the threshold (for example 150), it will take the value 255 (white), and if its value is lower (for example 100), it will take the value 0 (black). Apply an automatic/adaptive thresholding method and compare the obtained threshold with the suggested threshold. Plot the image histogram and analyze the automatic threshold in relation to the histogram's configuration. Add a comment in the code regarding this analysis.

Avec une image en couleur, on fera de même avec les trois composantes rouge, vert et bleu. Il y aura ainsi huit couleurs possibles pour chaque pixel : blanc, noir, rouge, vert, bleu, magenta, jaune et cyan.

With a color image, we will do the same with the three components red, green and blue. There will thus be eight possible colors for each pixel: white, black, red, green, blue, magenta, yellow and cyan.

7. "**Flip**". Produire une symétrie axiale d'axe horizontal.

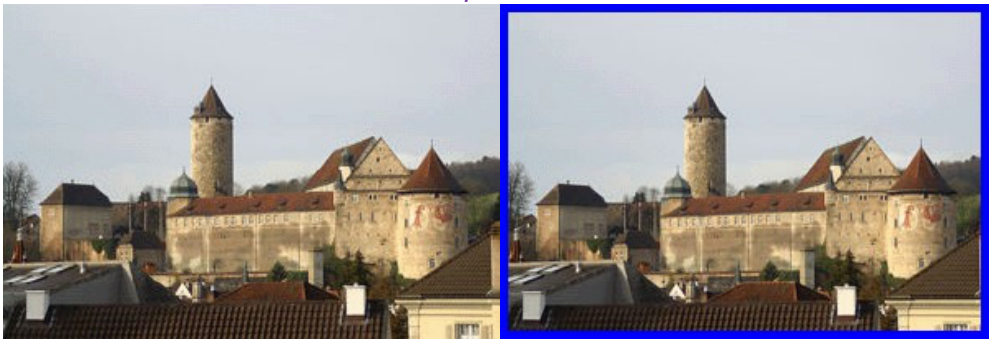
   Flip the image around the horizontal axis.



Retournement - Flip

8. "**Border**" : créer un bord bleu d'une largeur de 5 pixels.

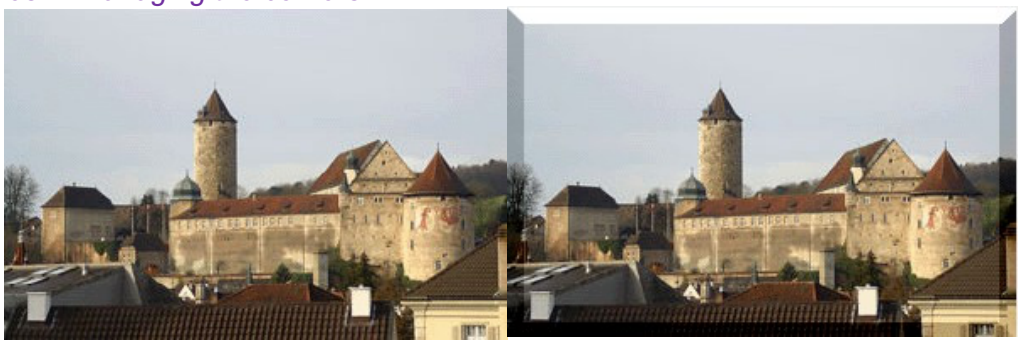
   Generate a blue border with a width of 5 pixels.



Bord - border

9. **"Relief"** : Créer un bord qui donnera une impression de relief. Il sera constitué de 4 régions polygonales d'une largeur de 10 pixels. Les pixels de la région du haut seront éclaircis de 65, ceux de la région du bas assombris de 65. A droite et à gauche, ils seront assombris de 40. La difficulté de l'exercice réside dans la gestion des coins...

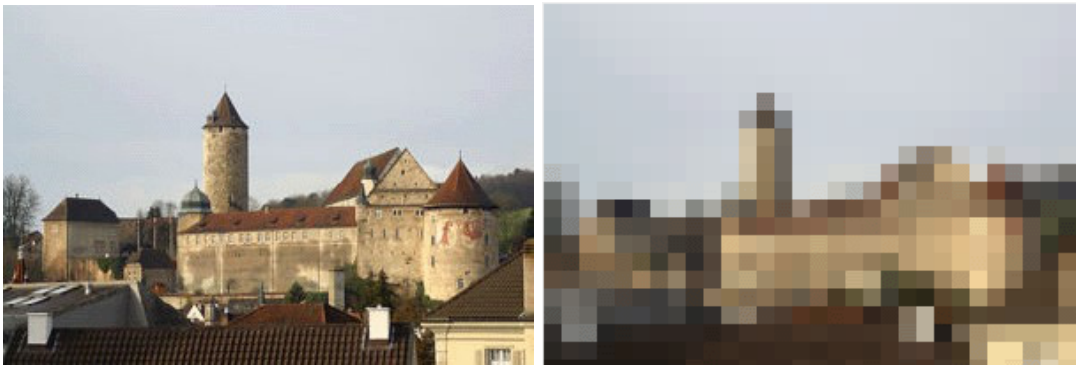
   Create an edge that will give an impression of relief. It will consist of 4 polygonal regions with a width of 10 pixels. The pixels of the top region will be lightened by 65, those of the bottom region darkened by 65. On the right and left, they will be darkened by 40. The difficulty of the exercise lies in managing the corners …



Relief

10. **"Pixeliser"** (x10). L'idée est de diviser l'image en carrés de 10 pixels sur 10 (dans les bords, ce seront des rectangles). Dans un "carré", chaque composante de chaque pixel sera la valeur moyenne du carré.

   "Pixelize" *(ordinal = 10). The idea is to divide the image into squares of 10 by 10 pixels (in the edges, they will be rectangles). In a "square", each component of each pixel will be the average value of the square.*



Pixélisation – Pixelize

11. **Lisibilité.** Que doit-on faire pour augmenter la lisibilité de l'image aquitain.tif (dans le dossier du TP) ? Mettre en œuvre vos différentes suggestions dans le cadre d'un script (Python).

   Readability. What should we do to increase the readability of the image aquitain.tif (in lab's folder)? Implementing your various suggestions within a (Python) script.

12. **Filtrage.** Le principe du filtrage est de modifier la valeur des pixels d'une image, généralement dans le but d'améliorer son aspect. En pratique, il s'agit de créer une nouvelle image en se servant des valeurs des pixels de l'image d'origine. Un filtre est une transformation mathématique (appelée produit de convolution) permettant de modifier la valeur d'un pixel en fonction des valeurs des pixels avoisinants, affectées de coefficients. Les calculs sont faits pour chacune des trois composantes de couleur. Le filtre est représenté par un tableau (une matrice), caractérisé par ses dimensions et ses coefficients, dont le centre correspond au pixel concerné. Habituellement, la somme des coefficients fasse 1, afin d'obtenir une valeur habituelle du niveau de gris résultant, quelle que soit la situation initiale (NB : il est possible de travailler avec des filtres ne vérifiant pas cette propriété mais le résultat devra être plafonné – il y aura donc une perte d'information). Rappelons que les valeurs des composantes des pixels sont des nombres entiers compris entre 0 et 255. Si les nouvelles valeurs ne sont plus des entiers, il faudra les arrondir.

Filtering. Filtering modifies the pixel values of an image, usually to improve its appearance. A new image is computed from the original by applying a *filter*, i.e. a mathematical transformation (convolution) that replaces each pixel with a weighted combination of its neighbours. The filter is defined by a matrix (kernel) whose size and coefficients determine the transformation. The sum of the coefficients is often set to 1 to preserve overall brightness, though other choices are possible (with potential loss of information). Since pixel components are integers between 0 and 255, computed values must be rounded.

a. Lissage. Le tableau ci-dessous rend l'image plus floue. On dit que c'est un filtre passe-bas. Appliquer ce tableau revient en fait à remplacer la valeur de chaque pixel par la moyenne des 9 pixels formant un carré autour du pixel visé. Implémenter et tester ce filtre.

Smoothing. The table below makes the image more blurred. This is an example of low pass filter. Applying this kernel *consists* in replacing the value of each pixel by the average of the 9 pixels forming a square around the targeted pixel. Implement and test this filter.

1/9	*1/9*	*1/9*
1/9	*1/9*	*1/9*
1/9	*1/9*	*1/9*

b. Accentuation. À l'inverse, le tableau ci-après rendra l'image plus nette. C'est un filtre passe-haut. Attention ! Il peut arriver que la nouvelle valeur ne soit plus comprise entre 0 et 255. Il faudra donc toujours prendre min(x, 255) et max(x, 0), où x est la nouvelle valeur. Implémenter et testez ce filtre.

Accentuation. At the opposite, the table below will make the image sharper. This is an example of high pass filter. Remark: it may happen that the new value is no longer between 0 and 255. It will therefore always be necessary to take min (x, 255) and max (x, 0), where x is the new value. Implement and test this filter.

0	-0.5	0
-0.5	3	-0.5
0	-0.5	0

c. ***Gradient*** *(filtre de Sobel). Pour faire simple, l'opérateur calcule le gradient de l'intensité de chaque pixel. Ceci indique la direction de la plus forte variation du clair au sombre, ainsi que le taux de changement dans cette direction. On connaît alors les points de changement soudain de luminosité, correspondant probablement à des bords. Implémentez ce filtre et testez-le.*

Gradient (Sobel filter). To *make* it simple, the operator calculates the gradient of the intensity of each pixel. This indicates the direction of the greatest change from light to dark, as well as the rate of change in that direction. We then know the points of sudden change in brightness, probably corresponding to edges. Implement and test this filter.

-1	0	1
-2	0	2
-1	0	1