Master 1 Informatique

Traitement d'image – 2023

TD2 – Operateurs de morphologiques et contours

Le but de ce TD est de manipuler et traiter une image à partir d'une bibliothèque OpenCV en Python. Nous allons appliquer les techniques suivantes sur les images :

- Opérations morphologiques sur l'image.
- Détection du model dans l'image.
- Calcule d'un gradient d'image.
- Détection des coins dans une image.

Exercice 1. Operateurs de morphologie mathématique

Utilisez les images `morph_1.pgm, morph_2.pgm, morph_3.pgm`

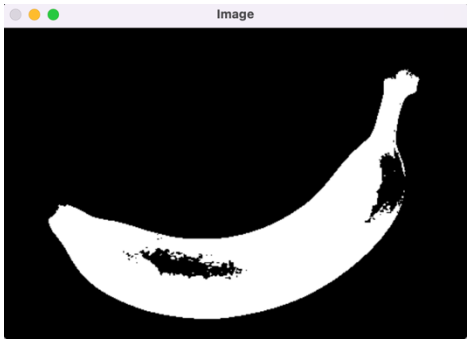
1) Prenez une image, seuillez cette image en testant plusieurs valeurs de seuils. Modifiez la fonction de seuillage afin que le fond de l'image soit en blanc et que les pixels des objets soient en noir. Comparer l'image seuillée avec au moins 3 valeurs différentes et en déduire le seuil le plus pertinent.
2) A partir de l'image seuillée la plus intéressante, écrire la fonction `erosion()`, qui va permettre de supprimer les points objets isolées. Exécuter cette fonction en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant.
3) A partir de l'image seuillée la plus intéressante, écrire la fonction `dilatation()`, qui va permettre de boucher des petits trous isolés dans les objets de l'image. Exécuter cette fonction en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant.

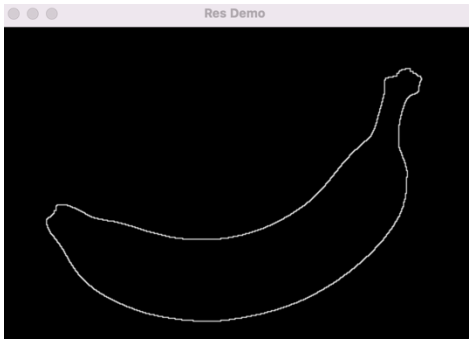
Utilisez les images `banane.jpg, tel.jpg`

4) Utilisez des fonctions `erode()` et `dilate()` d'OpenCV pour calculer le gradient morphologique. Testez-le sur différentes images.
   Le but de cet exercice est de développer une approche permettant de visualiser les contours d'une image. A partir de l'image érodée et de l'image dilatée, visualiser les contours des objets contenus dans l'image :
   - Si les deux pixels (de l'image érodée et de l'image dilatée) appartiennent au fond, alors le pixel correspondant de l'image de sortie appartiendra au fond (255).

- Si les deux pixels (de l'image érodée et de l'image dilatée) appartiennent à l'objet, alors le pixel correspondant de l'image de sortie appartiendra au fond (255).
- Si le pixel de l'image dilatée appartient à l'objet et que le pixel de l'image érodée appartient au fond, alors le pixel correspondant de l'image de sortie appartiendra au contour (0).

5) Comment corriger des erreurs de binarisation dans l'image ci-dessous et obtenir le contour d'objet « propre » ?





Pour plus d'information sur les opérations morphologique, regardez la documentation de OpenCV : https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

Exercice 2. – Où est Charlie ?

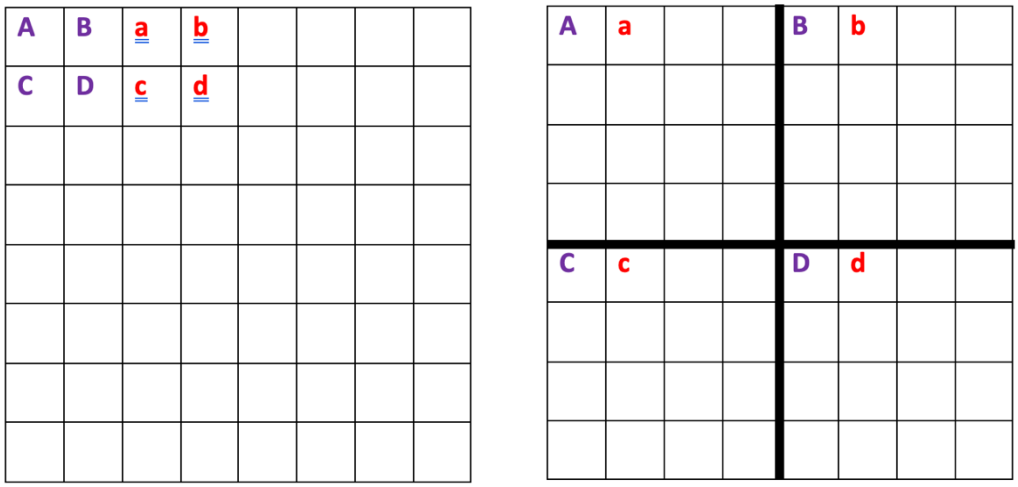
Utilisez la technique de détection du modèle dans l'image pour retrouver Charlie (`template.jpg`) dans l'image (`source.jpg`). Implémentez la fonction de la corrélation croisée normalisée.
Testez des autres modèles (`template2.jpeg` et `template3.jpg`) et des autres images de sources (`source2.jpeg` et `source3.jpg`). Que constatez-vous ?

Indice exercice 2 :
Regardez la fonction `matchTemplate()` dans le documentation de OpenCV https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html

Exercice 3. – Transformation du photomaton

On part d'un tableau $n \times n$, avec n pair, chaque élément du tableau représente un pixel. À partir de cette image on calcule une nouvelle image en déplaçant chaque pixel selon une transformation, appelée transformation du photomaton. On découpe l'image de départ selon des petits carrés de taille 2×2. Chaque petit carré est donc composé de quatre pixels. On envoie chacun de ces pixels à quatre endroits différents de la nouvelle image : le pixel en haut à gauche reste dans une zone en haut à gauche, le pixel en haut à droite du petit carré, est envoyé dans une zone en haut à droite de la nouvelle image, etc. (voir illustration).

Algorithme de transformation du photomaton :
Pour chaque couple (i, j), on calcule son image (i', j') par la transformation du photomaton selon les formules suivantes :
⋆ Si l'indice k est impair alors $k' = \frac{k+1}{2}$
⋆ Si l'indice k est pair alors $k' = \frac{k+n}{2}$,
Où k est soit i, soit j.



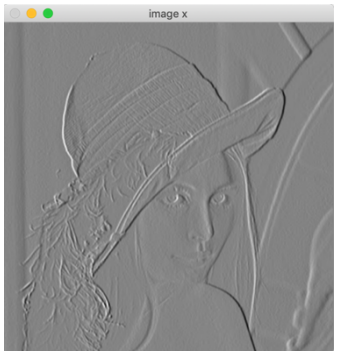
1) Programmez la transformation du photomaton d'une image de taille 256x256.
2) Faites-en sorte que votre programme fonctionne avec n'importe quelle image qui a le nombre pair de lignes et de colonnes.

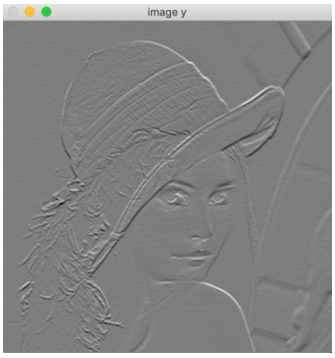
Référence : https://images.math.cnrs.fr/Mona-Lisa-au-photomaton.html

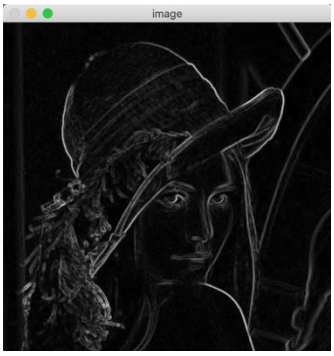
Exercice 4. – Gradient d'image

Calculez le gradient d'une image en direction x et y et détecter des contours en utilisant le détecteur de Sobel.

Indice : vous pouvez utiliser la fonction `Sobel()` du OpenCV.







Exercice 5. – Détection des coins Harris

Utilisez l'image `chessboard00.png`.

1) Détectez des coins dans l'image `chessboard00.png` en utilisant la fonction implémentée dans OpneCV `cornerHarris()`.

Vous pouvez utiliser le tutoriel du OpenCV :
https://docs.opencv.org/3.4/d4/d7d/tutorial_harris_detector.html

A quoi correspond la variable `thresh` ? Changez le code donné dans le tutoriel pour qu'on puisse définir par nous-même la valeur de la variable `thresh`. Quelle valeur devons-nous donner à cette variable pour retrouver tous les coins ?

Travail en autonomie :

2) Implémentation de la méthode de Harris pour la détection des coins dans une image sans utilisation de la fonction `cornerHarris()`.

Dans cet exercice, nous allons implémenter l'algorithme de la détection des coins Harris. Cette méthode est basée sur le gradient d'image. Une zone d'intérêt telle qu'un coin ou un bord correspond à un grand changement d'apparence. La détection des coins de Harris applique une approche mathématique à cette propriété afin de détecter les coins.
Pour chaque pixel, les points de coins peuvent être estimées à l'aide des valeurs propres de la matrice d'autocorrélation `M`, définies par :

$$M = \begin{pmatrix} \sum_W \langle I_x^2 \rangle & \sum_W \langle I_x I_y \rangle \\ \sum_W \langle I_x I_y \rangle & \sum_W \langle I_y^2 \rangle \end{pmatrix} \quad (1)$$

$$\langle I_x^2 \rangle = g \otimes I_x^2 \quad (2)$$

où W est un voisinage 3x3 autour du pixel, g un filtre gaussien et $\otimes$ l'opérateur de convolution.
Les valeurs propres de la matrice $M\ (\lambda_1, \lambda_2)$ donnent une classification des points d'image. Si λ_1 et λ_2 sont petits, la zone est plate. Si l'une des valeurs propres est très petite par rapport à l'autre, alors la zone correspond à une arête. Enfin, si λ_1 et λ_2 sont tous les deux grands et de même amplitude, la zone est une région d'angle.
Harris et Stefens ont suggéré d'utiliser le déterminant et la trace de M pour détecter les coins, ce qui élimine le besoin de calculer des valeurs propres :

$$R = \det(M) - k \cdot tr(M)^2 \qquad (3)$$

où k est une constante (avec une valeur typique de 0,04), $\det\ (M) = \lambda_1\lambda_2$ est le déterminant et $tr(M) = \lambda_1 + \lambda_2$ est la trace d'une matrice carrée.
R dépend uniquement des valeurs propres de M. R est grand pour un coin, R est négatif avec une grande amplitude pour un bord et |R| est petit pour une région plate.
La mise en œuvre de la méthode est la suivante :

- Calculez le gradient d'image pour chaque pixel dans les directions x et y :

$$I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}$$

- Calculez I_x^2, I_y^2 et $I_x I_y$.
- Lissez la dérivée de l'image carrée avec un filtre gaussien : $\langle I_x^2 \rangle = g \otimes I_x^2$.
- Pour chaque pixel, calculez la matrice d'autocorrélation M.
- Enfin, calculez R.

Si la matrice R est directement utilisée pour la détection des coins, chaque zone d'intérêt peut être détectée plusieurs fois.

Original image | Harris detection: R matrix | Result after NMS

Les redondances sont généralement corrigées à l'aide d'un algorithme de « suppression non maximale » (en anglais, *non-maximal suppression*), qui

consiste à mettre un pixel à zéro s'il y a une valeur plus élevée dans son voisinage. Un seuil peut être appliqué pour améliorer la détection.

a) Lisez l'image, calculez les dérivées d'image I_x et I_y et appliquez le filtre de lissage.
b) Calculez la matrice E qui contient pour chaque pixel la valeur de la plus petite valeur propre de M. Affichez la matrice E.
c) Calculez la matrice R qui contient pour chaque pixel le résultat de l'équation (3). Quelle est la différence avec E ?
d) Pour E et R, sélectionnez les 81 points les plus saillants. Obtenez-vous le résultat que vous attendiez ?
e) Appliquer un algorithme de suppression non maximal sur E et R (en choisissant un voisinage de dimension $11 * 11$), et afficher à nouveau les 81 points les plus saillants. Ce résultat est-il meilleur que le précédent ?

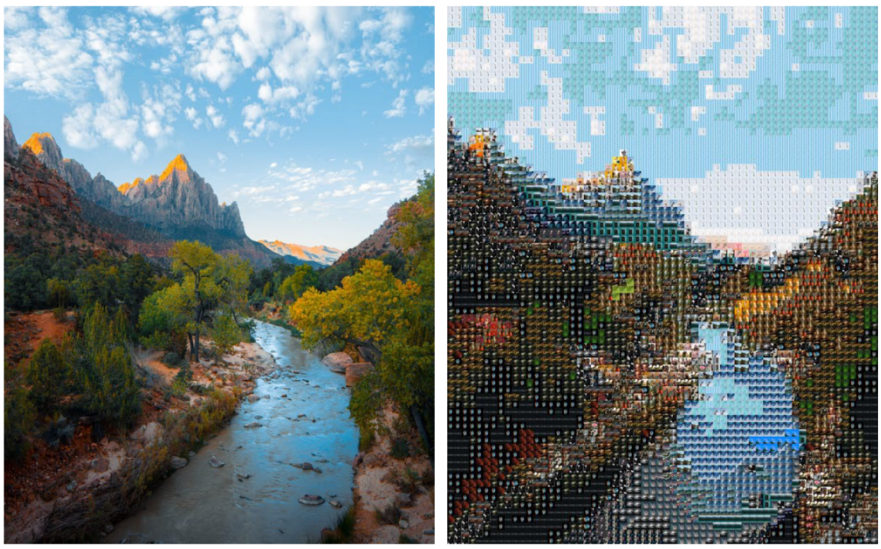
Exercice 6. – Mosaic d'image

Pour créer une photomosaïque, nous avons besoin de la photo principale et d'une liste de photos de tuiles. Les photos de tuiles doivent être de couleurs et de niveaux de luminosité différents pour de meilleurs résultats.
Ensuite, la procédure est la suivante :

1. Pixéliser la photo principale.
2. Remplacez chaque pixel par la photo de mosaïque la plus proche en couleur.

Indice :
Pour calculer la couleur la plus proche, utilisez la distance Euclidienne :

$$d = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}$$



Original photo by Michael Block from Pexels (left), photo mosaic (right) (All photos from Pexels)