Filière : API2

Module : Python

Traitement d'images

Réaliser par :

AIT TOUDA Omar

AMROUCH Aicha

AKILO ILLA Abdou Razak

ALHAJ OUSMANE Abdelsalam

Encadré par :

Pr. SAADI Mostafa

Année Universitaire : 2023-2024

Table des matières

Remerciement

Cher Professeur,

Nous tenons tout d'abord à vous remercier pour votre encadrement et votre soutien durant la réalisation de notre mini projet sur le traitement d'image en Python. Votre disponibilité et votre expertise ont été précieuses pour nous aider à comprendre les différentes techniques de traitement d'image et à mettre en pratique nos connaissances.

Introduction

Le traitement d'images est une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information. Il s'agit d'un sous-ensemble du traitement du signal dédié aux images et aux données dérivées comme la vidéo (par opposition aux parties du traitement du signal consacrées à d'autres types de données : son et autres signaux monodimensionnels notamment), tout en opérant dans le domaine numérique (par opposition aux techniques analogiques de traitement du signal, comme la photographie ou la télévision traditionnelles).

Dans le contexte de la vision artificielle, le traitement d'images se place après les étapes d'acquisition et de numérisation, assurant les transformations d'images et la partie de calcul permettant d'aller vers une interprétation des images traitées.

CHAPITRE 1 : Implémentation

Partie 1 : les opérations d'entrée/sortie sur les images

Les opérations d'entrée/sortie (E/S) sur les images en Python permettent de lire et d'écrire des fichiers image dans différents formats, tels que JPEG, PNG et BMP. Elles sont importantes lors du traitement et de l'analyse d'images numériques en Python :

Il existe plusieurs bibliothèques Python qui offrent des fonctionnalités d'E/S d'images, telles que OpenCV, Pillow ,matplotlib et scikit-image. Ces bibliothèques permettent de lire et d'écrire des fichiers image, ainsi que de les afficher à l'écran ou de les enregistrer dans un fichier.

Voici un exemple de lecture et d'affichage d'une image en utilisant la bibliothèque Pillow en Python :

```
import matplotlib.pyplot as plt
from PIL import Image
def AfficherImg(img):
 plt.axis("off")
 plt.imshow(img ,cmap = "gray", interpolation="nearest")
#plt.imshow(img, cmap = "gray")#palette predefinie pour afficher une image
 plt.show()
################## Ouvrir une image jpg ou bmp on retourne une matrice
def ouvrirImage(chemin):
 img=plt.imread(chemin)
 return img
```

Afin d'enregistrer une image sur le disque dur, on utilise la fonction Image.save() , qui prend en argument cette image soit sous forme jpg ou bmp.

```python
import matplotlib.pyplot as plt
################ Sauvgarder une image sous forme jpg ou bmp
def saveImage(img):
    plt.imsave("image1.jpg",img)
```

Partie 2 : Les Images Noires et Blanches

Créer une image noire : Pour créer une image noire de hauteur h et de largeur l on utilise une fonction qui prend en paramètres deux entiers et retourne une matrice contenant h lignes et l colonnes dont la valeur de chaque pixel est initialisé par 0.

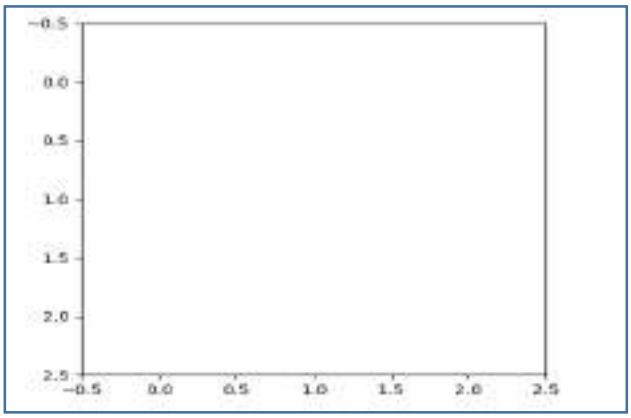
```python
def image_noire(h, l):
    # Créer une matrice h x l avec des valeurs initiales de 0
    matrice_image = [[0] * l for _ in range(h)]

    return matrice_image
```

Créer une image blanche : Pour générer une image en blanc de hauteur h et de largeur l, on utilise la fonction d'entête def image _blanche (h,l) qui retourne une matrice contenant h ligne et l colonnes dont la valeur de chaque pixel et initialisé par le nombre 1

```python
def image_blanche(h, l):
    # Créer une matrice h x l avec des valeurs initiales de 1
    matrice_image = [[1] * l for _ in range(h)]

    return matrice_image
```

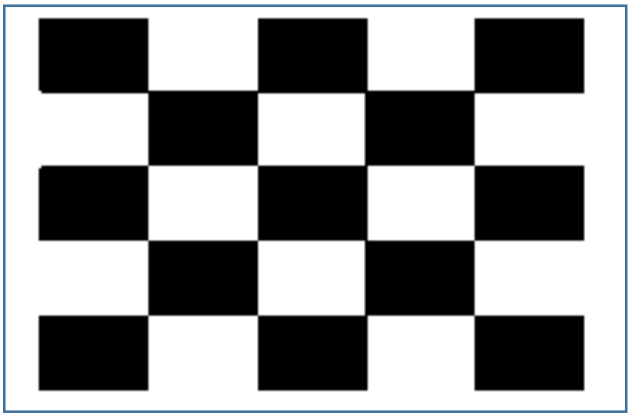



```python
plt.axis("on")
```

pour afficher les axes sur image

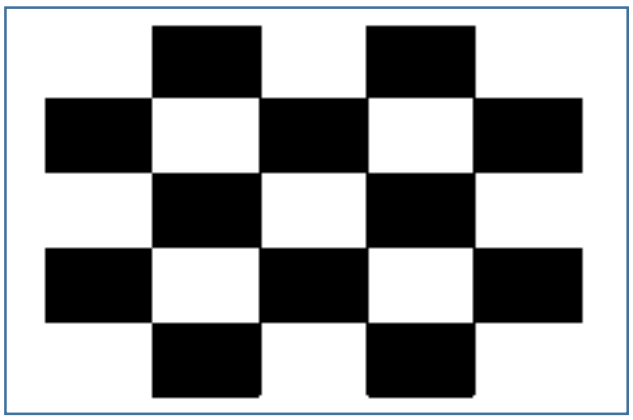
Créer une image en Noir et Blanc : La fonction suivante s'appelle def creelmgBlancNoir(h,l) qui retourne une image noir et blanc dont le contenu de chaque pixel (i,j) est donné par (i+j)%2.

```python
def creerImgBlancNoir(h, l):
    Mbn=[[(i+j)%2 for j in range (l)]for i in range (h)]
    return Mbn
```



Créer image négatif : La fonction def negatif(Img) prend en entrée une matrice Img et renvoie le négatif de l'image d'entrée. Le négatif d'une image est obtenu en inversant les couleurs de l'image originale, c'est-à-dire en remplaçant chaque pixel de l'image par son opposé dans le spectre de couleurs. Par exemple, un pixel blanc sera remplacé par un pixel noir et un pixel noir sera remplacé par un pixel blanc. Cette fonction est souvent utilisée en traitement d'image pour mettre en évidence certains détails de l'image qui ne sont pas facilement visibles dans l'image originale.

```
def negatif(img_negatif):
        # Inverser les valeurs de la matrice
    for i in range(len(img_negatif)):
        for j in range(len(img_negatif[0])):
            if img_negatif[i][j]==0:
                img_negatif[i][j]=1
            else :
                img_negatif[i][j]=0
    return img_negatif
```



Partie 3 : Les Images en niveau de gris :

La première fonction, luminance, retourne la luminance d'une image, c'est-à-dire la moyenne de tous les pixels de l'image. La deuxième fonction, contraste,

retourne le contraste d'une image, c'est-à- dire la variance des niveaux de gris de l'image. La troisième fonction, profondeur, retourne la valeur maximale d'un pixel dans l'image. Enfin, la quatrième fonction, Ouvrir, retourne la matrice représentant l'image passée en argument.

```python
def luminance(img):
    somme = 0
    for i in range(len(img)):
        for j in range(len(img[0])):
            # calculer la somme de tous les pixels
            somme += img[i][j]
    return somme / (len(img) * len(img[0]))
def contrast(img):
    ctrst = 0
    N = len(img) * len(img[0])
    avg_luminance = luminance(img)
    for i in range(len(img)):
        for j in range(len(img[0])):
            ctrst += (img[i][j] - avg_luminance) ** 2
    return ctrst / N
```

```python
# Q11
def profondeur(Img):
    max= Img[0][0]
    for pixel in Img :
        for ligne in pixel:
            if ligne > max :
                max = ligne
    return max
# Q 12
def Ouvrir(Img):
    # plt.imread()  est une fonction dans pyplot qui retourn la matrice d'une image
    Img=plt.imread(Img)
    return Img
```

Partie 4 : Les opérations élémentaires sur les images en mode gris :

La fonction inverser (img) : La fonction prend en entrée une matrice "img" représentant une image sous forme de liste bidimensionnelle. Elle crée une nouvelle matrice appelée "invrs" de la même taille que l'image d'origine, initialisée avec des zéros. Ensuite, la fonction parcourt chaque élément de la matrice d'entrée et remplace sa valeur par son complément à un (1 - img[i][j]). En d'autres termes, elle inverse les valeurs binaires de l'image, transformant les pixels noirs en blancs et vice versa. Ainsi, la fonction retourne une nouvelle matrice qui représente l'image inversée. Ce processus est couramment utilisé dans le domaine du traitement d'images pour effectuer des opérations de négatif sur une image.

. Par exemple, l'image B résulte de l'application de l'inverse sur l'image A de l'introduction .

```
def inverser(img):
    # cree une matrice initialisé par des zeros
    invrs=[[0]*len(img[0]) for e in range (len(img))]
    for i  in range (len(img)) :
        for j in range (len(img[0])):
            invrs[i][j]=1- img[i][j]
    return invrs
```

Image A

Image B

La fonction flipH(img) : La fonction 'filpH' semble être conçue pour effectuer une opération de retournement horizontal sur une image représentée

sous forme de liste de listes (matrice). Elle crée une nouvelle matrice appelée de ‘flip_H’ , même taille que l'image d'entrée ‘img’, et remplit chaque élément de cette matrice en copiant les éléments correspondants de la ligne inverse de l'image d'origine.
L’image C résulte de l’application de la fonction ‘ flipH ’ sur l’image A de l’introduction.

```
def flipH(img):
    flip_H =[[0]*len(img[0]) for e in range (len(img))]
    for i  in range (len(img)) :
        for j in range (len(img[0])):
            flip_H[i][j]= img[i][-j-1]
    return flip_H
```

La fonction poserV(img1, img2) : La fonction ‘ poserV ’ prend deux images en entrée, ‘ img1 ’ et ‘ img2 ’, et génère une nouvelle image, ‘ imgV ’, en réalisant une addition pixel par pixel entre les deux images d'entrée. Chaque pixel de l'image résultante est obtenu en additionnant les valeurs correspondantes des pixels dans les images initiales. Ce processus peut être utilisé pour diverses applications dans le domaine du traitement d'images, telles que la fusion d'images ou d'autres opérations où l'addition élément par élément des pixels est nécessaire.

```python
def poserV(img1, img2):
    # l'addition de deux images est réalisée élément par élément
    imgV = img1 + img2
    return imgV
```

La fonction poserH(img1, img2) : cette fonction prend deux listes d'images, ‘ img1 ’et ‘ img2 ’, en tant que paramètres. Ces images sont représentées sous forme de listes de lignes, où chaque ligne contient des éléments représentant les pixels ou les valeurs d'intérêt. La fonction parcourt ensuite chaque ligne des deux images, additionne les éléments correspondants de chaque ligne, et crée ainsi une nouvelle ligne dans une nouvelle image résultante, que l'on nomme ‘ imgH ’. En d'autres termes, la fonction réalise une opération de concaténation horizontale des deux images en additionnant les éléments pixel par pixel. La nouvelle image résultante contiendra donc les informations combinées des deux images d'origine, alignées horizontalement .

```python
def poserH(img1, img2):
    # dans cette liste on parcourir chaque ligne des deux images
    # et ajouter les éléments correspondants de chaque ligne
    # pour créer une nouvelle ligne dans la nouvelle image
    imgH = [img1[i] + img2[i] for i in range(len(img1))]
    return imgH
```

POSER
IMAGE
HORIZONTAL

Partie 5 : Les images RGB

Question 22) Les valeurs des expressions sont :

```
50
255
255
```

Question 23) Donner la quantité de mémoire nécessaire en octets(8 bits) pour stocker le tableau représentant une image RGB :

Pour stocker une image RGB (Red Green Blue), vous avez besoin de trois valeurs de couleur par pixel. Chaque valeur de couleur est généralement représentée sur 8 bits, ce qui signifie qu'elle peut prendre un nombre compris entre 0 et 255.

Donc, pour stocker une image RGB, vous avez besoin de 3 * 8 = 24 bits par pixel. Si votre image a une largeur de W pixels et une hauteur de H pixels, vous avez besoin de W * H * 24 bits de mémoire pour stocker l'image complète.

En utilisant l'unité de mesure des octets (8 bits), cela signifie que vous avez besoin de W * H * 3 octets de mémoire pour stocker une image RGB.

Initialisation d'une image RGB : Voici comment vous pouvez écrire une fonction pour initialiser et renvoyer un tableau imageRGB à trois dimensions de manière aléatoire en utilisant la fonction ' randrange( ) ' :

```
def initImageRGB(imageRGB):
    # initial chaque pixel de l' image avec  une valeur du couleur aléatoire
    for i in range (len(imageRGB) ) :
        for j in range (len(imageRGB[0]) ):
            imageRGB[i][j]=(randrange(255),randrange(255),randrange(255))
            # renvoyer le tableau image RGB aléatoire
    return imageRGB
```

Pour tester cette fonction, vous pouvez créer un tableau imageRGB vide et appeler la fonction ' initImageRGB() ' sur ce tableau. Par exemple :

```
#  TSET THE FUNCTION
#  créeons d'abord un tableau vide de 3*3
imageRGB=[[0 for j in range (3) ] for i in range (3)]
imageRGB=initImageRGB(imageRGB)
print(imageRGB)
```

Voici comment vous pouvez écrire une fonction qui lit une image à partir d'un fichier, la convertit en une matrice de valeurs de couleur, calcule sa symétrie par rapport à l'axe horizontal et convertit le résultat en une image réelle

```
def symetrieH(img):
    img_symtrH =[[0]*len(img[0]) for e in range (len(img))]
    for i  in range (len(img)) :
        for j in range (len(img[0])):
            img_symtrH[i][j]= img[i][-j]
    return img_symtrH
```

La fonction "symetrieH" prend en entrée une matrice représentant une image et crée une nouvelle matrice qui résulte de la symétrie horizontale de l'image d'origine. Elle initialise une matrice vide appelée "img_symtrH" et remplace chaque pixel de cette matrice en le copiant depuis la colonne opposée de l'image d'entrée. Ainsi, la fonction produit une version symétrique horizontale de l'image d'origine, reflétant chaque pixel par rapport à l'axe vertical central.

Résultat :

Image ***originale***

Image ***symétrie***

Voici comment vous pouvez écrire une fonction qui lit une image à partir d'un fichier, la convertit en une matrice de valeurs de couleur, calcule sa symétrie par rapport à l'axe vertical et convertit le résultat en une image réelle

```
def symetrieV(img):
    img_symtrV =[[0]*len(img[0]) for e in range (len(img))]
    for i  in range (len(img)) :
        for j in range (len(img[0])):
            img_symtrV[i][j]= img[len(img) - 1 - i][j]
    return img_symtrV
```

Cette fonction fonctionne de manière similaire à la fonction précédente, mais elle calcule la symétrie par rapport à l'axe vertical en parcourant chaque pixel et en copiant sa valeur dans la position symétrique de l'image symétrique.

Résultat :

Image ***originale***

Image ***symétrie***

La conversion d'une image RGB en une image de niveau de gris se présente comme ceci:

- Chercher le maximum et le minimum pour chaque pixel sur les trois couches.
- Calculer la partie entière de la moyenne de ces valeurs maximales et minimales.
- Obtenir la valeur du nouveau pixel en niveaux de gris.

```
def grayscale(imageRGB):
    #### creation d'une novelle matrice
    img_Gris=[[0]*len(imageRGB[0]) for e in range(len(imageRGB))]
    for i in range(len(imageRGB)):
        for j in range(len(imageRGB[0])):
            elet_max=max(imageRGB[i][j])
            elet_min=min(imageRGB[i][j])
            moy=elet_min # ou moy=elet_min
            img_Gris[i][j]=moy
    return img_Gris

# test the function
ImgRGB=ouvrirImage(r"c:\Users\PC\OneDrive\Bureau\WhatsApp Image .jpg")
matrice_de_gris=grayscale(ImgRGB)
AfficherImg(matrice_de_gris)
```

CHAPITRE 2 : Quelques exemples d'utilisation

- Le traitement d'image est une technique largement utilisée en intelligence artificielle pour analyser et interpréter les images numériques. Il permet de mettre en œuvre des algorithmes pour extraire des informations utiles à partir d'images, comme la reconnaissance de formes, la détection de mouvement ou la mesure de distance.

- Le traitement d'image est souvent utilisé dans les domaines de la vision par ordinateur, de la robotique et de la reconnaissance de formes, où il est nécessaire de traiter de grandes quantités de données visuelles de manière efficace. Il peut également être utilisé dans d'autres domaines, comme la médecine, l'astronomie ou la sécurité, pour analyser des images médicales, des images de l'univers ou des images de caméras de surveillance.

- Pour traiter les images, les algorithmes utilisent souvent des techniques de filtrage, de transformation et de segmentation, qui permettent de nettoyer, de transformer et de séparer les différentes parties de l'image. Ils peuvent également utiliser des techniques de reconnaissance de formes, comme la reconnaissance de caractères ou de visages, pour identifier des éléments spécifiques dans l'image.

- En utilisant ces techniques, les algorithmes de traitement d'image peuvent extraire des informations précieuses à partir des images, comme la localisation des objets, la détection des mouvements ou la mesure de distances. Ces informations peuvent être utilisées pour améliorer la précision

des systèmes de vision par ordinateur, pour naviguer de manière autonome dans des environnements complexes ou pour détecter des anomalies dans les images.

CHAPITRE 3 : L'utilisation de traitement des images dans la vie de l'ingénieur

Le traitement d'images est un domaine important de l'ingénierie, qui a de nombreuses applications dans la vie quotidienne. Les ingénieurs qui travaillent dans le traitement d'images peuvent être impliqués dans des projets tels que :

Amélioration de la qualité des images : les ingénieurs peuvent utiliser le traitement d'images pour améliorer la qualité des images, par exemple en enlevant le bruit ou en corrigeant les distorsions.

Analyse d'images : les ingénieurs peuvent utiliser le traitement d'images pour extraire des informations utiles à partir d'images, par exemple en détectant des objets ou en mesurant des caractéristiques d'une image.

Création d'images : les ingénieurs peuvent utiliser le traitement d'images pour créer des images à partir de données brutes, par exemple en générant des images de synthèse à partir de modèles 3D.

Bilan qualitatif

Le projet a été réalisé dans le but de développer une solution de traitement d'images en Python. Il a été structuré en plusieurs étapes afin de rendre la compréhension et l'utilisation de la solution aussi simple que possible.

Les fonctions du projet ont été soigneusement documentées et des exemples d'utilisation ont été fournis pour faciliter l'intégration de la solution dans d'autres projets.

En résumé, ce projet a permis de mettre en place une solution de traitement d'images en Python complète et efficace, qui peut être facilement intégrée et utilisée dans d'autres projets.

ETAPES	DESCRIPTION
1	Objectif : Solution de traitement d'images en Python
2	Fonctions : traitement
3	Documentation et exemples
4	Intégration dans d'autres projets

Ce tableau récapitule les différentes étapes et caractéristiques du projet de traitement d'images en Python de manière claire et concise. Il permet de comprendre facilement l'objectif et les fonctionnalités de la solution développée.

Conclusion

Le traitement d'image est un sujet de grande importance dans le monde de l'informatique, et l'utilisation de Python pour cette tâche est devenue de plus en plus fréquente ces dernières années. En effet, Python est un langage de

programmation facile à apprendre et à utiliser, qui offre de nombreuses bibliothèques dédiées au traitement d'image. Ces bibliothèques permettent de réaliser des opérations de traitement d'image de manière simple et rapide, ce qui en fait un outil précieux pour les développeurs et les scientifiques travaillant dans ce domaine. Grâce à ses capacités de traitement d'image, Python est utilisé dans de nombreux projets de recherche et de développement, ainsi que dans l'industrie pour la création de logiciels de traitement d'image de haute qualité. En somme, l'utilisation de Python pour le traitement d'image est un choix judicieux pour ceux qui cherchent à obtenir des résultats de qualité de manière efficace

Bibliographie

Le site officiel python : https://www.python.org

Le site Wikipedia : https://fr.m.Wikipédia.org

Le site officiel : https://pixees.fr/informatiquelycee/n_site/snt_photo_transImg.html

Le site officiel python développez : https://python.developpez.com/faq/?page=Pil

Le site officiel openclassrooms : https://openclassrooms.com/fr/courses/4470531classez-et- segmentez-des-donnees-visuelles/5024566-appliquez-vos-premierstraitements-dimages