Sciences Numériques et Technologie	2nde
Thème 6 : LA PHOTOGRAPHIE NUMÉRIQUE Activité 3 : les algorithmes de traitement des photographies numériques	TP

Contenus	Capacités attendues
Traitement d'image.	Traiter par programme une image pour la transformer en agissant sur les trois composantes de ses pixels.

Restitution de l'activité sous forme de compte rendu que vous rédigerez avec un logiciel de traitement de texte.

Le compte-rendu présentera votre travail question par question. Selon l'exercice, soit vous indiquez les réponses aux questions, soit vous collez votre code python (une capture d'écran suffit) et l'image obtenue après traitement.

1. Analyse de quelques algorithmes

Durée estimée : 15 minutes

La qualité des photographies prises par les appareils photo numériques ou les smartphones augmente d'année en année.

Il devient de plus en plus facile de réaliser une photographie qui satisfait nos attentes. Si des progrès ont eu lieu dans le domaine de l'optique, c'est essentiellement aux progrès fulgurants des algorithmes de traitement d'images que l'on doit la satisfaction d'une photographie réussie.

Les algorithmes présentés ci-dessous peuvent être utilisés en post-traitement de photographie (sur un ordinateur avec un logiciel dédié), par le biais d'un filtre appliqué sur un réseau social, ou même de manière automatique lors de la prise de vue, lorsque ces algorithmes sont implémentés dans l'appareil photo numérique.

1.1. Algorithme 1 : Fusion automatique de plusieurs photographies pour ne garder que des visages souriants.

1.2. Algorithme 2 : Effet Bokeh ⇒ modification artificielle de la profondeur de champ.

Appelé « mode Portrait » sur iOS

1.3. Algorithme 3 : Focus stacking ⇒ plusieurs photos de profondeurs de champs différentes sont fusionnées pour que le premier plan et l'arrière-plan soient nets en même temps.

1.4. Algorithme 4 : correction de la distorsion

Exercice à réaliser dans le compte rendu de cette activité :

1. Classez les algorithmes présentés ci-dessus dans un tableau en indiquant leur numéro.

Algorithmes essayant de reproduire le plus fidèlement possible une réalité	Algorithmes essayant d'imiter un effet artistique de la photo argentique	Algorithmes produisant une photo d'une situation qui n'a jamais existé

2. Citez et décrivez les algorithmes (filtres) que vous utilisez le plus souvent.

2. Manipulation d'images avec Python

Créez un dossier **SNT_manip_image**. Nous travaillerons dans ce dossier.

Logiciel : Thonny
Modules à installer dans Thonny : pillow

Fichiers nécessaire pour réaliser le TP : *fleur.jpg* - *deathstar.png* - *ciel-bleu.jpg*

ATTENTION : il faut cliquer sur le bouton télécharger (en haut à droite) et enregistrer les 3 photos dans le dossier → **SNT_manip_image**

2.1. Comment passer une image en niveau de gris ?

Durée estimée : 25 minutes

Les codes et les images produites doivent être placés dans votre compte rendu.

Dans cet exercice, vous allez réaliser un programme pour appliquer un filtre qui permet de faire passer l'image en niveau de gris.

- Enregistrez l'image ***fleur.jpg*** dans le dossier **SNT_manip_image.**
- Le programme à créer dans cette partie doit être enregistré dans le même dossier et nommé : ***VotreNOM-fleur.py***

Une façon simple de passer une image en couleurs en niveaux de gris est de remplacer le triplet (R,V,B) par la moyenne arithmétique de R, V et B.
Exemple : un pixel coloré dont le triplet RVB vaut (150, 150 , 0) sera remplacé par (100, 100, 100)
calcul : (150 + 150 + 0) / 3 = 100

- Copiez le programme présenté en page 3 et complétez les lignes 10, 11, 12 et 13.
  Pour cela, le programme doit:
  - calculer la moyenne des intensités de couleur du pixel traité ;
  - convertir le nombre obtenu en entier grâce à la fonction `int()` ;
  - affecter cette valeur aux trois intensités de couleurs du pixel de la nouvelle image.

```
from PIL import Image

def traitement(image):
    # Dimensions de l'image
    largeur, hauteur = image.size
    image_traitee = Image.new('RGB', (largeur, hauteur))
    for y in range(hauteur) :
        for x in range (largeur) :
            R, V, B = image.getpixel((x,y))
            Moyenne = int (.............................)
            R_traite = ..................................
            V_traite = ..................................
            B_traite = ..................................
            # Modification du pixel de l'image
            image_traitee.putpixel((x, y), (R_traite, V_traite, B_traite))
    return image_traitee

# Ouverture de l'image
image_source = Image.open("fleurs.jpg")
image_traitee = traitement(image_source)

# sauvegarde de l'image_source
image_traitee.save("fleur-gris.jpg", "JPEG")
image_traitee.show()
```

Facultatif : modifier votre programme pour obtenir une nouvelle image « fleur-gris2.jpg » en vous référant aux indications ci-dessous.

*Bonus : une autre façon de transformer une image en niveau de gris consiste à remplacer la moyenne précédente par la valeur de ce calcul : 0,2125 * R + 0,7154 * V + 0,0721 * B*

2.2. Comment fusionner deux images ?

Durée estimée : 35 minutes

Les codes et les images produites doivent être placés dans votre compte rendu.

Vous avez déjà vu des tournages vidéo réalisés devant un fond vert. En post-production, les pixels verts sont remplacés par des pixels d'une autre vidéo, donnant une incrustation parfaite du personnage dans un nouveau décor. Dans cet exercice, nous allons réalisé un programme python pour réaliser une opération similaire sur deux images.

L'objectif de cet exercice est de remplacer le fond vert d'une image par le paysage de la deuxième image.
Les deux images ont la même définition : 512 x 387.

- Enregistrez les images *deathstar.png* et *ciel-bleu.jpg* dans le dossier **SNT_manip_image.**
- Le programme à créer dans chaque partie devra être enregistré dans le même dossier.

2.2.a *1ère partie du programme : analyse de la 1ère image*

Enregistrez votre programme avec le nom : *detourage1.py*

On ouvre l'image et on lui attribue le nom `img1`. Dans toute la suite du programme, elle sera désignée par `img1`

```
from PIL import Image

img1=Image.open("deathstar.png")
p=img1.getpixel((0,0))
print (p)
```

Quelle valeur renvoie la commande `print (p)` (regardez la console) ? Est-ce cohérent ?

Coup de pouce : l'image est au format.png. Ce format gère la transparence ce qui doit rajouter une valeur au triplet RVB.

Ajouter ces deux lignes à la fin du programme

```
6.    p2=img1.getpixel((100,60))
7.    print (p2)
```

Quelles sont les valeurs RVB du pixel de coordonnées (100,60) ?

Ajouter ces lignes à votre code qui permette d'afficher « le pixel ... est vert » si la valeur respective des pixels « p » et « p2 » observés correspond à celle d'un pixel vert.

```
8.    if p==(0,255,0,255):
9.       print ("le pixel p est vert")
10.   if p2==(0,255,0,255):
11.      print ("le pixel p2 est vert")
```

Pensez à enregistrer votre travail.

2.2.b 2ème partie du programme : parcourir tous les pixels de l'image

Enregistrez votre programme avec le nom : *detourage2.py*

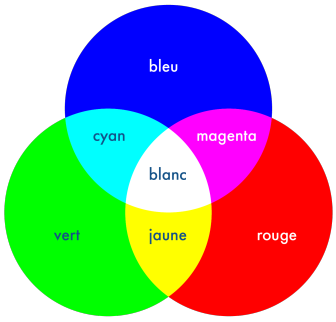
À l'aide d'une double boucle, nous allons parcourir tous les pixels de l'image et renvoyer le message « le pixel est vert » lorsque l'on rencontre un pixel vert.

Effacez les lignes 4 à 11 et remplacez les par :

```
4.  for x in range (512):
5.     for y in range (387):
6.        p=img1.getpixel((x,y))
7.        if p==(0,255,0,255):
8.           print ("ce pixel est vert")
9.        else :
10.          print ("autre couleur")
```

Exécutez ce script et observez la console.

Remarque : vous pouvez interrompre votre code si son exécution totale prend trop de temps.



2.2.c 3ème partie du programme : parcourir tous les pixels de l'image

Enregistrez votre programme avec le nom : *detourage3.py*

Effacez les lignes 9 et 10
Modifiez votre code précédent afin que tous les pixels verts deviennent magenta.
Il suffit de remplacer print ("ce pixel est vert") par `img1.putpixel ((x,y),(…,…,…,…))`

N'oubliez pas d'enregistrer l'image à la fin du code.
`img1.save("deathstar-magenta.png", "PNG")`

Remarque : le détourage est imparfait, il reste des pixels verts en bordure de l'étoile. Cela vient du fait qu'il y a des nuances de vert en bordure de l'étoile. Ouvrez l'image, zoomé et observez les pixels...

2.2.d 4ème partie du programme : fusionner deux images

Enregistrez votre programme avec le nom : *detourage4.py*

En s'inspirant des programmes précédents, remplacez chaque pixel vert par le pixel situé au même endroit dans l'image *ciel-bleu.jpg (img2)*.
Aide : vous vous souvenez de ➔ `img1.putpixel((x,y),q)`

N'oubliez pas d'enregistrer l'image à la fin du code.
`img1.save("deathstar-ciel-bleu.png", "PNG")`

Remarque : il est possible de réduire les résidus de vert en modifiant la ligne 8 en ⇒ `if p[0]<100 and p[1]>150 and p[2]<100:`

```
1   from PIL import Image
2
3   img1=Image.open("deathstar.png")
4   img2=Image.open("ciel-bleu.jpg")
5   for x in range (512):
6     for y in range (387):
7       p=img1.getpixel((x,y))
8       if p==(          ):
9         q=img2.getpixel((x,y))
10        ((x,y),q)
11
12  img1.save("deathstar-ciel-bleu.png")
```