Traitement d'images (Partie 2)

Objectifs

- comprendre la notion de langages compilés ou interprétés
- découvrir le mécanisme de gestions des exceptions du langage Python
- savoir appliquer un filtre à une image
- savoir construire l'histogramme des intensités de couleurs d'une image

Connaissances et savoir-faire à maîtriser

1) Langage compilé vs langage interprété

Lorsque vous exécutez votre programme Python, plusieurs opérations totalement transparentes pour l'utilisateur sont effectuées :

- une phase d'**analyse syntaxique** qui vérifie si les règles grammaticales du langage sont respectés. C'est lors de cette phase que vous pouvez être confronté à des erreurs de type *Invalid Syntax* ;
- une phase de production d'un code objet dit **bytecode** ;
- une phase d'exécution de ce bytecode par l'**interpréteur** Python.

On dit que Python est un langage interprété car le bytecode généré a l'avantage de pouvoir être exécuté sur n'importe quel système (*Windows*, *Linux* ou *MacOS*) pour peu que l'utilisateur dispose de l'interpréteur Python adapté à son système. On dit que Python est un **langage portable**.

*Remarque 1 : à l'inverse, un langage de programmation est dit **compilé** lorsque le code source, autrement dit le programme que vous avez écrit, est traduit en un code binaire directement exécutable par la machine. Le code produit n'est pas portable : il ne peut être exécuté que par le système d'exploitation sur lequel il a été compilé. On ne pourra pas par exemple lancer un exécutable compilé sous Windows sous un environnement MacOS.*

Remarque 2 : chaque langage possède ses avantages et ses inconvénients. Un programme écrit avec un langage interprété (Python, Java) est généralement plus facile à mettre en oeuvre mais moins rapide à l'exécution que le même programme écrit en langage compilé (C, C++).

2) La gestion des exceptions

Lors de la phase d'interprétation du *bytecode Python*, une erreur (on parle d'**exception**) peut-être levée, le programme s'arrête alors brusquement. Par exemple, lorque l'utilisateur saisit des caractères non numériques que vous cherchez à transtyper en un entier, Python lève une exception de type `ValueError` et le programme s'arrête.
Il existe un mécanisme en Python qui permet d'**attraper** ou d'**intercepter** ces exceptions afin de les gérer proprement et ainsi faire en sorte que le programme continue à s'exécuter normalement. Pour cela, on utlise les instructions suivantes :

```
try:
    nb = int(input("Saisir un entier : "))
except ValueError:
    print("ERREUR : le nombre saisi n'est pas un entier!!!")
```

Plus d'informations : https://docs.python.org/fr/3/tutorial/errors.html

BY-NC-ND Mathieu Pons mathete.net

3) Filtres de couleurs

Nous avons déjà vu comment convertir une image en noir et blanc ou en niveau de gris ou comment filtrer les couleurs d'une image. Il existe d'autres filtres que l'on obtient en jouant avec la valeur des canaux de couleurs comme l'illustrent les images suivantes :

FIGURE 1 – Négatif

FIGURE 2 – Permutation

FIGURE 3 – Modulation

FIGURE 4 – Luminosité ↓

FIGURE 5 – Original

FIGURE 6 – Luminosité ↑

a. Négatif d'une image

le négatif d'une image est obtenu simplement en affectant à chaque canal x d'un pixel la valeur $255 - x$:

```
img.putpixel((x, y), (255 - r, 255 - g, 255 - b))
```

b. Permutation de couleurs

On permute les couleurs d'une image en changeant l'ordre des valeurs de chaque composante :

```
img.putpixel((x, y), (b, r, g))
```

c. Modulation de couleurs

La modulation de couleur fait subir à la valeur d'une ou plusieurs composantes une transformation numérique, par exemple :

```
img.putpixel((x, y), (5 * r % 256, g, b))
```

Le symbole % se nomme l'opérateur **modulo**. Il permet d'obtenir le reste de la division euclidienne de $5 \times r$ par 256. Ainsi, si $r = 123$ alors $5 \times r = 615$ et `615 % 256` renvoit 103 car $615 = 2 \times 256 + 103$, autrement dit 103 est le reste de la division euclidienne de 615 par 256. On s'assure ainsi que la valeur renvoyée est un entier dans l'intervalle $[0\,;255]$.

BY-NC-ND Mathieu Pons mathete.net

Remarque 3 : l'opérateur modulo est très utilisé en mathématiques et en informatique et intervient entre autres dans les procédés de cryptage d'informations.

d. Variation de luminosité

On augmente ou on abaisse la luminosité générale d'une image en ajoutant ou en retranchant à chaque composante de couleur des pixels une valeur fixe.

```
img.putpixel((x, y), (r + 50, g + 50, b + 50)) # hausse de luminosité
img.putpixel((x, y), (r - 20, g - 20, b - 20)) # baisse de luminosité
```

Il vous faudra toutefois veiller à mettre en place un système permettant de vérifier si la valeur est toujours dans l'intervalle $[0\,;255]$. Si ce n'est pas le cas, les valeurs inférieures à 0 sont fixées à 0, celles supérieures à 255 sont fixées à 255.

Remarque 4 : une augmentation trop importante de la luminosité va avoir pour conséquence de tirer l'image vers le blanc puisque les pixels déjà clairs deviendront totalement blanc.

e. Augmentation de contraste

Le contraste mesure la différence de luminosité entre les zones claires et les zones sombres d'une image. Pour l'augmenter, il suffit d'éclaircir les pixels "clairs" et d'assombrir les pixels "sombres". Il est alors nécessaire de "décider" si tel ou tel pixel est considéré comme clair ou sombre : on pourra pour cela calculer la luminance du pixel comme vu dans le TP précédent et la comparer à un seuil fixé.

Remarque 5 : l'image obtenue avec ce procédé est peu satisfaisante car en réalité les algorithmes réhausseur de contraste sont beaucoup plus élaborés.

4) Histogramme des couleurs

Dans la plupart des logiciels de retouche d'images, il existe une fonction permettant d'afficher l'histogramme des couleurs d'une image. Celui-ci représente la distribution des intensités de chaque couleur dans l'image. Pour construire un tel graphique, il faut suivre les étapes suivantes :

1. créer trois listes `histo_red`, `histo_green`, `histo_blue` de 256 éléments initialisées chacun à la valeur 0. On pourra pour cela utliser un raccourci syntaxique permis par le langage Python et qui évite l'utilisation d'une boucle FOR :

```
histo_red = [0] * 256
# équivaut à ...
for i in range(256):
    histo_red.append(0)
```

2. pour chaque pixel de l'image, prélever la quantité de rouge, de vert et de bleu contenu dans ce pixel via son code RGB et incrémenter d'une unité sa valeur correspondante dans le tableau. Par exemple, si le code RGB prélevé est `(123, 56, 201)`, `histo_red[123]` est augmenté de 1, de même que `histo_green[56]` et `histo_blue[201]` indiquant qu'il existe dans l'image un pixel supplémentaire dont la quantité de rouge est 123, celle de vert est 56 et celle de bleu 201 ;
3. tracer l'histogramme de chaque canal de couleur avec un module approprié au tracé de courbes.

BY-NC-ND Mathieu Pons mathete.net

Pour le tracé nous utiliserons la bibliothèque `matplotlib` :

```
from matplotlib import pyplot

# Mise en place du graphique
pyplot.title("Histogramme du rouge") # titre du graphique
pyplot.xlabel("Intensités du rouge") # label sur l'axe des abscisses
pyplot.ylabel("Nombre de pixels") # label sur l'axe des ordonnées

# Diagramme en barres OU tracé de la courbe
pyplot.bar(range(256), histo_red, color = "red") # affiche des barres
pyplot.plot(histo_red, color = "red") # affiche une courbe
pyplot.show() # affichage
```

Pour notre image de perroquet, les histogrammes de distribution des intensités de chaque couleur sont les suivants :

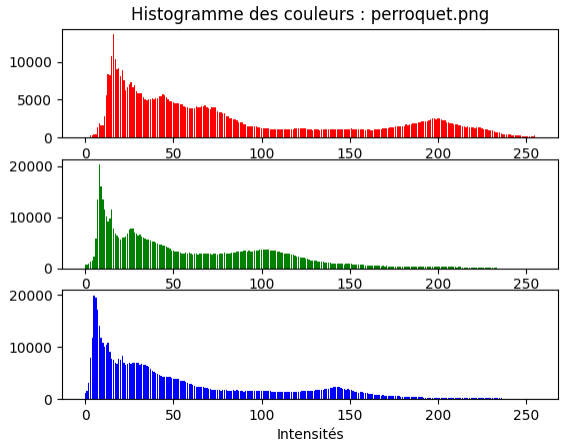

FIGURE 7 – Par canal, en barres

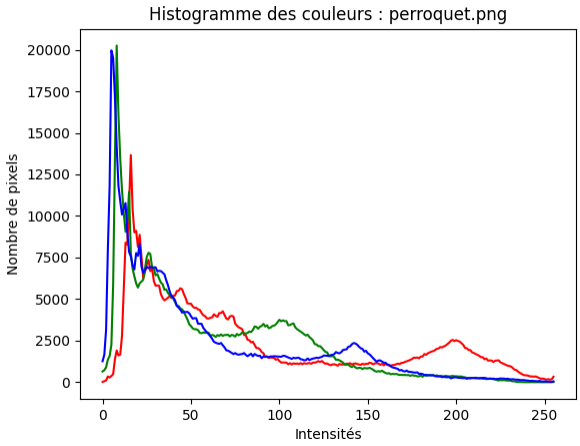

FIGURE 8 – Tracé unique

Remarque 6 : `matplotlib` est une bibliothèque très complète utilisée pour l'affichage de données scientifiques, vous trouverez davantage d'informations à son sujet à l'adresse suivante : https://matplotlib.org/

Exercices à réaliser

Vous pouvez travailler sur l'image retaillée dans le TP précédent `perroquet.png` ou tout autre image de votre choix. Toutefois, veillez à disposer d'une image dont les dimensions sont raisonnables car le parcours de l'image pixel par pixel est coûteux en terme de performance.

EXERCICE 1

Ouvrir le fichier `TP-images-partie2-exo1.py` et répondre aux questions suivantes :

1. La fonction `afficher_infos` du TP précédent souffre de certaines lacunes. Elle lève notamment une exception si l'image passée en paramètre est une image créée par vos soins :

```
img = Image.new("RGB", (100, 100))
afficher_info(img)
# A l'exécution, Python mentionne que img n'a pas de nom
AttributeError: 'Image' object has no attribute 'filename'
```

BY-NC-ND Mathieu Pons mathete.net

Implémenter un mécanisme de gestion d'exceptions permettant de résoudre cette problématique.

2. Écrire une fonction `afficher_histogramme(pimg)` qui affiche l'histogramme des couleurs d'une image tel que représentées sur les figures 7 et 8.

Exercice 2

L'objet de cet exercice est de créer un programme qui, à partir d'une image de votre choix, applique les différents filtres détaillés dans ce TP. Si vous les testez sur notre perroquet, vous devriez obtenir les images des figures 1 à 6.

Compétences à valider

Traitement d'images (II)		☼	⛅	🌧	⛈
Théorie	Je sais faire la différence entre langage compilé et langage interprété				
	J'ai compris le mécanisme de gestions des exceptions du langage Python				
Pratique	Je sais construire le négatif d'une image				
	Je sais appliquer un filtre de permutation ou de modulation des couleurs d'une image				
	Je sais augmenter ou abaisser la luminosité globale d'une image				
	Je sais construire l'histogramme des intensités de couleurs d'une image				
	Je parviens à organiser mon code de manière propre et suffisamment documentée				

BY-NC-ND Mathieu Pons mathete.net