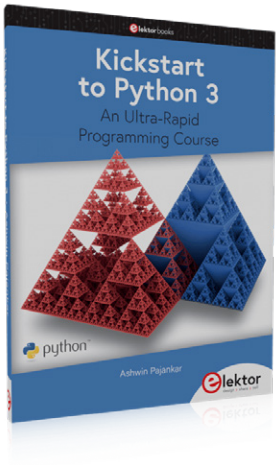
Bibliothèque Wand et traitement d'images

extrait du livre *Kickstart to Python 3*



Ashwin Pajankar (Pakistan)

Ashwin Pajankar, auteur du livre d'Elektor « Kickstart to Python 3», se penche dans cet article sur un domaine où Python se classe parmi les meilleurs langages de programmation : le traitement d'images. Rejoignez Ashwin pour explorer une bibliothèque de traitement d'images nommée Wand, qui se distingue par sa capacité de programmation des effets spéciaux en toute simplicité et rapidité, bien entendu en Python.

Note de l'éditeur : *cet article est un extrait du livre* **Kickstart to Python 3** *(Elektor, 2022), formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor. Puisque cet article est extrait d'une publication plus vaste, certains termes peuvent faire référence à des passages du livre d'origine situés ailleurs. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions – pour les contacter, voir l'encadré* **« Des questions, des commentaires ? ».**

Le traitement d'images implique l'utilisation d'algorithmes pour traiter le contenu des images. À l'époque des films analogiques et des images animées, il existait des procédés pour améliorer la qualité des images et des trames (dans une image animée) avec des techniques manuelles telle que l'utilisation de composés chimiques. C'était le prélude à la notion moderne de traitement d'images. Aujourd'hui, la plupart des images sont numériques. Bien entendu, le numérique n'a pas encore rattrapé les couleurs vives et la clarté de l'analogique (imagerie chimique sur film). Cependant, une grande majorité de personnes et d'organisations (de production et de traitement de films) utilisent l'imagerie numérique dans la production d'images et de vidéos puisqu'elle revient moins chère. Les ordinateurs modernes sont également suffisamment rapides pour le traitement des images numériques. En général, nous utilisons des langages de programmation modernes comme C, C++, Java, Python, MATLAB et GNU Octave pour traiter les images et les vidéos. Python simplifie le traitement des images grâce aux nombreuses bibliothèques tierces disponibles.

ImageMagick est un logiciel pour la manipulation des images. Il comprend des API pour différents langages de programmation. Nous pouvons utiliser la bibliothèque **Wand** qui fournit une interface Python à ImageMagick. Commençons par installer les logiciels nécessaires. Nous devons d'abord installer ImageMagick pour nos systèmes d'exploitation. Nous pouvons installer ImageMagick sur macOS avec les commandes suivantes :

```
brew install ghostscript
brew install imagemagick
```

Ces deux commandes permettent d'installer ImageMagick sur votre macOS. Mais si vous ne réussissez pas, vous devez l'installer manuellement. C'est simple - téléchargez le fichier zip trouvé à l'adresse [1] et copiez-le dans le répertoire personnel de votre utilisateur sous macOS. Extrayez-le avec la commande suivante :

```
tar xvzf ImageMagick-x86_64-apple-darwin20.1.0.tar.gz
```

Nous devons maintenant ajouter quelques entrées au fichier *.bash_profile* qui se trouve dans le répertoire personnel de votre utilisateur sous macOS.

```
# Settings for ImageMagik
export MAGICK_HOME="$HOME/ImageMagick-7.0.10"
export PATH="$MAGICK_HOME/bin:$PATH"
export DYLD_LIBRARY_PATH="$MAGICK_HOME/lib/"
```

Quittez et relancez l'invite de commande et exécutez les commandes suivantes une par une :

```
magick logo: logo.gif
identify logo.gif
display logo.gif
```

qui afficheront le logo du projet ImageMagick.

L'installation sous Windows est simple. Il existe des fichiers binaires exécutables à installer pour toutes les versions de Windows de bureau (32/64 bits). Parmi toutes les options, nous devons choisir celle avec la description *Win64/Win32 dynamic at 16 bits-per-pixel component with High-dynamic-range imaging enabled*. Pour les systèmes 64 bits, utilisez [2] et pour Windows 32 bits, utilisez [3].

Les utilisateurs de Linux peuvent télécharger le fichier source avec

Figure 1. Image affichée dans le Jupyter notebook.

la commande suivante,

```
wget https://www.imagemagick.org/download/ImageMagick.
tar.gz
```

Vérifions où tous les fichiers ont été extraits :

```
ls ImageMagick*
```

Cela nous indique le nom du répertoire :

```
ImageMagick-7.1.0-10
```

Accédez au répertoire :

```
cd ImageMagick-7.1.0-10
```

Ensuite, exécutez les commandes suivantes l'une après l'autre (si vous êtes familier avec Linux, vous saurez qu'il s'agit d'un ensemble standard de commandes pour installer tout nouveau programme sur les distros Linux) :

```
./configure
make
sudo make install
sudo ldconfig /usr/local/lib
```

Après avoir installé avec succès le programme ImageMagick, nous pouvons installer la bibliothèque Wand sur n'importe quelle plateforme avec la commande suivante :

```
pip3 install wand
```

Ceci achève l'installation d'ImageMagick et de Wand sur n'importe quel système d'exploitation.

Pour commencer

Veuillez créer un nouveau Jupyter *notebook* pour toutes les démonstrations de cet article. À partir de ce stade, tout le code doit être enregistré et exécuté dans le *notebook*. Commençons par importer les bibliothèques nécessaires.

```
from __future__ import print_function
from wand.image import Image
```

Ces commandes importent les modules nécessaires. Lisons une image et imprimons ses dimensions comme suit :

```
img = Image(filename='D:/Dataset/4.2.03.tiff')
print('width =', img.width)
print('height =', img.height)
```

le résultat est comme suit :

```
width = 512
height = 512
```

Nous pouvons également voir le type d'image :

```
type(img)
```

Ceci donne le résultat suivant :

```
wand.image.Image
```

Ensuite, nous pouvons afficher l'image dans le *notebook* comme sortie en tapant simplement le nom de la variable qui stocke l'image :

```
img
```

Ceci crée la sortie montrée dans la **figure 1**.

J'utilise l'ensemble de données d'images fourni par ImageProcessingPlace [4]. Toutes les images sont des images de test standard fréquemment utilisées en traitement d'images. Je n'utilise pas l'image de test Lena, car je pense que son origine est controversée et qu'elle est dévalorisante et irrespectueuse envers les femmes en général.

Nous pouvons également cloner une image, modifier son format de fichier et l'enregistrer sur le disque comme suit :

```
img1 = img.clone()
img1.format = 'png'
img1.save(filename='D:/Dataset/output.png')
```

Si vous ne l'avez pas encore remarqué, j'utilise un ordinateur Windows pour cette démo. Si vous utilisez un autre système d'exploitation Unix, vous devez en conséquence modifier l'emplacement. Par exemple, j'utilise le code suivant pour enregistrer le fichier de sortie sur un ordinateur Raspberry Pi OS (version Debian Linux) :

```
img1.save(filename='/home/pi/Dataset/output.png')
```

Nous pouvons également créer une image personnalisée avec une couleur uniforme :

```
from wand.color import Color
bg = Color('black')
img = Image(width=256, height=256, background=bg)
img.save(filename='D:/Dataset/output.png')
```

Voici comment redimensionner une image. Il y a deux façons, la première :

```
img = Image(filename='D:/Dataset/4.2.03.tiff')
img1 = img.clone()
img1.resize(60, 60)
img1.size
```

et la deuxième :

```
img1 = img.clone()
img1.sample(60, 60)
img1.size
```

Les routines `resize()` et `sample()` redimensionnent l'image aux valeurs spécifiées. Nous pouvons également rogner une partie de l'image, comme ceci :

```
img1 = img.clone()
img1.crop(10, 10, 60, 60)
img1.size
```

Figure 2. Image floue.

Figure 3. Flou adaptatif.

Figure 4. Flou de mouvement avec un angle de 30 degrés.

Figure 5. Flou de rotation.

Figure 6. Flou sélectif.

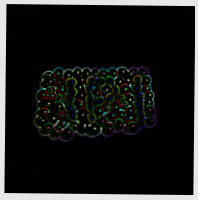


Figure 7. Détection des contours.

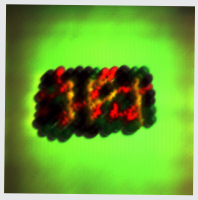


Figure 8. Effet de gaufrage.

Effets d'image

Nous disposons d'une grande variété d'effets d'image. Commençons par rendre une image floue.

```
img1 = img.clone()
img1.blur(radius=6, sigma=3)
img1
```

Le résultat est représenté sur la **figure 2**. Maintenant, appliquons un flou adaptatif :

```
img1 = img.clone()
img1.adaptive_blur(radius=12, sigma=6)
img1
```

et ensuite un flou gaussien :

```
img1 = img.clone()
img1.gaussian_blur(sigma=8)
img1
```

Le résultat est illustré à la **figure 3**. Nous pouvons également appliquer un effet de flou de mouvement :

```
img1 = img.clone()
img1.motion_blur(radius=20, sigma=10, angle=-30)
img1
```

Notez que nous indiquons l'angle de mouvement en appelant la routine. La sortie devrait être telle que dans la **figure 4**.

Ensuite, on utilise le flou de rotation (**figure 5**) :

```
img1 = img.clone()
img1.rotational_blur(angle=25)
img1
```

et le flou sélectif (**figure 6**) :

```
img1 = img.clone()
img1.selective_blur(radius=10, sigma=5,
threshold=0.50 * img.quantum_range)
img1
```

Il est également possible de débruiter une image, c'est-à-dire réduire le bruit :

```
img1 = img.clone()
img1.despeckle()
img1
```

et détecter les contours (**figure 7**) :

```
img = Image(filename='D:/Dataset/4.1.07.tiff')
img1 = img.clone()
img1.edge(radius=1)
img1
```

Nous pouvons générer un effet de gaufrage en 3D (**figure 8**) :

```
img1 = img.clone()
img1.emboss(radius=4.5, sigma=3)
img1
```

ou changer l'image en échelle de gris et appliquer un effet d'image (**figure 9**) :

```
img1 = img.clone()
img1.transform_colorspace('gray')
img1.emboss(radius=4.5, sigma=3)
img1
```

Pour une netteté globale de l'image, il suffit de programmer :

```
img1 = img.clone()
img1.sharpen(radius=12, sigma=4)
img1
```

d'appliquer l'algorithme de netteté adaptative :

```
img1 = img.clone()
img1.adaptive_sharpen(radius=12, sigma=6)
img1
```

ou en procédant à l'inverse en utilisant le filtre de suppression de la netteté :

```
img1 = img.clone()
img1.unsharp_mask(radius=20, sigma=5,
amount=2, threshold=0)
img1
```

Enfin, essayez de répartir les pixels de façon aléatoire dans le rayon spécifié :

```
img1 = img.clone()
img1.spread(radius=15.0)
img1
```

et observez le résultat comme dans la **figure 10**.

Effets spéciaux

Découvrons comment appliquer des effets spéciaux à une image. Le premier effet est le bruit. Il existe différents types de bruit. Voyons comment introduire un bruit gaussien.

```
img1 = img.clone()
img1.noise("gaussian", attenuate=1.0)
img1
```

La sortie est représentée à la **figure 11**. Voici une liste de toutes les chaînes de caractères valides qui peuvent être utilisées comme noms de types de bruit :

```
'gaussian'
```

```
'impulse'
'laplacian'
'multiplicative_gaussian'
'poisson'
'random'
'uniform'
```

Nous pouvons appliquer le décalage vers le bleu à une image de la manière suivante (**figure 12**) :

```
img1 = img.clone()
img1.blue_shift(factor=0.5)
img1
```

ou créer l'effet de dessin au fusain (**figure 13**) :

```
img1 = img.clone()
img1.charcoal(radius=2, sigma=1)
img1
```

Nous pouvons aussi appliquer une matrice de couleurs :

```
img1 = img.clone()
matrix = [[0, 0, 1],
[0, 1, 0],
[1, 0, 0]]
img1.color_matrix(matrix)
img1
```

Une matrice de couleurs peut avoir une taille maximale de 6 x 6. Dans une matrice de couleurs, chaque colonne correspond à un canal de couleur à référencer, et chaque ligne représente un canal de couleur à affecter. Pour les images RGB, il s'agit de rouge, vert, bleu, n/a, alpha et une constante (*offset*). Pour les images CMYK, il s'agit de cyan, jaune, magenta, noir, alpha et d'une constante. Dans cet exemple, nous avons créé une matrice 3 x 3 dont le résultat est illustré à la **figure 14**.

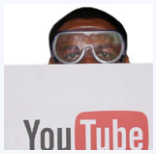
Le programme prend en charge d'autres effets spéciaux, décrits dans le livre, notamment le mélange avec une couleur constante, l'implosion, le Polaroid, la sépia, l'esquisse, la solarisation, le tourbillon, la teinte, la vignette, la vague, le débruitage par ondelettes, le flip, le flop et la rotation.

220314-04 – VF : Asma Adhimi

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (ashwin.pajankar@gmail.com) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur



Ashwin Pajankar est titulaire d'un MASTER en technologie de l'IIIT Hyderabad et a plus de 25 ans d'expérience en programmation. Il a commencé son parcours dans la programmation et l'électronique avec le langage de programmation BASIC et maîtrise maintenant la programmation en assembleur, C, C++, Java, Shell Scripting et Python. Il a également acquis une expérience technique dans le domaine des ordinateurs monocartes tels que Raspberry Pi, Banana Pro et Arduino.

Figure 9. Gaufrage sur une image en échelle de gris.

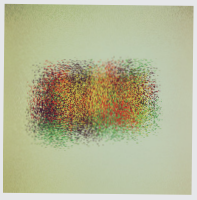


Figure 10. Effet d'étalement.

Figure 11. Bruit gaussien.

Figure 12. Image décalée vers le bleu.

Figure 13. Effet dessin au fusain.

Figure 14. Effet de matrice de couleurs.

LIENS

[1] Téléchargement d'ImageMagick :
https://download.imagemagick.org/ImageMagick/download/binaries/ImageMagick-x86_64-apple-darwin20.1.0.tar.gz
[2] ImageMagick pour les systèmes Win64 :
https://download.imagemagick.org/ImageMagick/download/binaries/ImageMagick-7.1.0-10-Q16-HDRI-x64-dll.exe
[3] ImageMagick pour les systèmes Win32 :
https://download.imagemagick.org/ImageMagick/download/binaries/ImageMagick-7.1.0-10-Q16-HDRI-x86-dll.exe
[4] Bases de données d'images, ImageProcessingPlace.com :
www.imageprocessingplace.com/root_files_V3/image_databases.htm