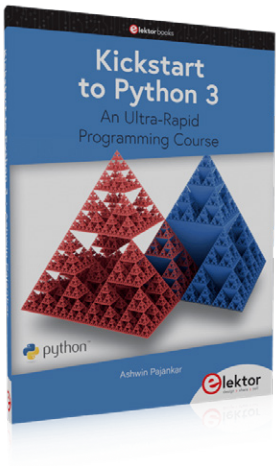
Snelstart met Python 3

voorbeeldhoofdstuk: digitale beeldbewerking en de Wand-bibliotheek



Ashwin Pajankar (Pakistan)

Ashwin Pajankar, Elektor boek-auteur, onderzoekt hier een gebied waar Python als programmeertaal hoog wordt aangeslagen, zo niet 'nummer één' is: beeldbewerking. Onderzoek samen met Ashwin de Wand-beeldverwerkingsbibliotheek, die uitblinkt door gemakkelijk programmeren van speciale effecten met minimale omhaal – en dat alles in Python, natuurlijk, en vooral snel!

Noot van de redactie: *dit artikel is een deel uit het 210 pagina's tellende boek* ***"Kickstart to Python 3"*** *(Elektor 2022). Het fragment is opgemaakt en licht geredigeerd om te voldoen aan de redactionele standaarden en pagina-indeling van Elektor Magazine. Als deel een grotere publicatie kunnen sommige termen in dit artikel verwijzen naar passages elders in het boek. Auteur en redactie hebben hun best gedaan om dat te vermijden, en staan voor u klaat bij eventuele vragen (zie het kader* ***Vragen of opmerkingen?****).*

'Beeldbewerking' is het gebruik van algoritmen om beeldinhoud te bewerken. In de tijd van analoge fotografie en speelfilms waren er technieken om de kwaliteit van beelden en frames (in een speelfilm) 'met de hand' te verbeteren met bijvoorbeeld chemische verbindingen. Dit was een voorloper van de moderne beeldbewerking. Tegenwoordig zijn de meeste beelden digitaal. Natuurlijk kan digitaal nog niet tippen aan de levendige kleuren en helderheid van analoog (beeldbewerking op basis van chemische film). Maar omdat het goedkoper is, gebruikt de overgrote meerderheid van mensen en organisaties (filmproducerende en -verwerkende organisaties) digitale beeldbewerking bij de productie van beelden en video's. Moderne computers zijn ook snel genoeg voor de bewerking van digitale beelden. Daarvoor wordt meestal gebruik gemaakt van moderne programmeertalen zoals C, C++, Java, Python, MATLAB en GNU Octave. Het is bijzonder gemakkelijk om beelden te bewerken met Python, omdat er veel bibliotheken van derden voor beschikbaar zijn.

ImageMagick is een softwaretool voor de manipulatie van beelden. Het wordt geleverd met API's voor verschillende programmeertalen. We kunnen de bibliotheek **Wand** gebruiken die een python-interface voor ImageMagick biedt. Laten we eerst de benodigde software installeren. Eerst moeten we ImageMagick installeren voor onze besturingssysteem. Onder macOS installeren we ImageMagick met de volgende commando's:

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

Deze twee commando's zouden ImageMagick moeten installeren onder macOS. Zo niet, dan moeten we het handmatig doen. Dat is eenvoudig – download het zip-bestand van [1] en kopieer het naar de *user home*-directory onder macOS. Pak het uit met het volgende commando:

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

Nu moeten we een paar regels toevoegen aan het bestand *.bash_profile* in de *user home*-directory onder 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/"
```

Sluit de opdrachtprompt af en start deze opnieuw,; voer dan de volgende commando's één voor één uit:

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

Nu verschijnt het logo van het ImageMagick-project.

Installatie onder Windows is eenvoudig. Er zijn binaire uitvoerbare installatiebestanden voor alle smaken van Windows (32/64 bit). Van alle mogelijkheden moeten we die kiezen met de beschrijving *Win64/Win32 dynamic at 16 bits-per-pixel component with High-dynamic-range imaging enabled*. Voor 64bit-systemen gebruik je [2] en voor 32bit-Windows gebruik je [3].

Figuur 1. Een afbeelding in het Jupyter Notebook.

Linux-gebruikers kunnen de broncode downloaden met het volgende commando:

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

Laten we eens kijken waar alle bestanden zijn uitgepakt:

```
ls ImageMagick*
```

We krijgen de naam van de directory:

```
ImageMagick-7.1.0-10
```

Ga naar die directory:

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

Voer vervolgens de volgende commando's na elkaar uit (als je bekend bent met Linux, zul je herkennen dat dit de standaardset commando's is om een nieuw programma op Linux-distro's te installeren):

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

Nadat het ImageMagick-programma succesvol is geïnstalleerd, kunnen we de Wand-bibliotheek onder elk platform installeren met het volgende commando:

```
pip3 install wand
```

Hiermee is de installatie van ImageMagick en Wand onder elk besturingssysteem voltooid.

Aan de slag

Maak een nieuw Jupyter Notebook aan voor alle voorbeelden in dit artikel. Vanaf hier moet alle code worden opgeslagen en uitgevoerd in dat Notebook. We beginnen met het importeren van de benodigde bibliotheken.

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

Deze statements importeren de vereiste modules. Laten we eerst een afbeelding inlezen en de afmetingen ervan afdrukken; dat doen we als volgt:

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

De uitvoer is als volgt:

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

We kunnen ook het type van de afbeelding opvragen:

```
type(img)
```

Dit geeft de volgende uitvoer:

```
wand.image.Image
```

Vervolgens kunnen we de afbeelding in het Notebook als uitvoer tonen door gewoon de naam van de variabele in te typen waarin de afbeelding is opgeslagen:

```
img
```

Dit produceert de uitvoer van **figuur 1**.
Ik gebruik de afbeeldings-dataset die wordt geleverd door Image-ProcessingPlace [4]. Alle afbeeldingen zijn standaard testafbeeldingen die vaak in beeldbewerking worden gebruikt. Ik gebruik niet de standaard-testafbeelding "Lena" omdat ik van mening ben dat de herkomst van de afbeelding controversieel is en hij respectloos en denigrerend is ten opzichte van vrouwen in het algemeen.
We kunnen ook een afbeelding klonen, het bestandsformaat veranderen en op schijf opslaan:

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

Als het je nog niet is opgevallen: ik gebruik een Windows-computer voor deze voorbeelden. Als je een Unix-achtig OS gebruikt, moet je de locatie navenant aanpassen. Ik gebruik bijvoorbeeld de onderstaande code om het uitvoerbestand op een Raspberry Pi OS (Debian Linux-variant) op te slaan:

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

We kunnen ook een aangepaste afbeelding maken met uniforme kleur:

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

Laten we eens kijken hoe we de afmetingen van een afbeelding kunnen veranderen. Er zijn twee manieren. De eerste:

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

En de tweede:

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

De routines `resize()` en `sample()` verkleinen de afbeelding tot de gespecificeerde afmetingen. We kunnen ook een deel van een afbeelding bijsnijden, zoals hier:

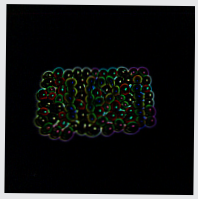
Figuur 2. Een onscherpe afbeelding.

Figuur 3. Adaptieve vervaging.

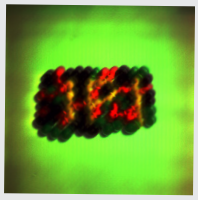
Figuur 4. Bewegings-onscherpte onder een hoek van 30°.

Figuur 5. Rotatievervaging.

Figuur 6. Selectieve vervaging.



Figuur 7. Randdetectie.



Figuur 8. Reliëfeffect.

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

Beeldeffecten

Er zijn veel verschillende beeldeffecten mogelijk. Laten we beginnen met het vervagen van een afbeelding.

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

De uitvoer is weergegeven in **figuur 2**. Laten we nu adaptieve vervaging toepassen:

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

en dan wat Gaussiaanse vervaging:

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

De uitvoer is te zien in **figuur 3**. We kunnen ook bewegingsonscherpte toepassen:

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

Merk op dat we de bewegingshoek opgeven bij de aanroep van de routine. De uitvoer moet er als **figuur 4** uitzien.
Dan is er nog de draaiings-onscherpte (**figuur 5**):

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

en selectieve onscherpte (**figuur 6**):

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

We kunnen ook een beeld 'ontspikkelen' (despeckle), dat wil zeggen de ruis reduceren:

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

en randen detecteren (**figuur 7**):

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

We kunnen een 3D-reliëfeffect genereren (**figuur 8**):

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

of het beeld naar grijstinten omzetten en er dan een effect op loslaten (**figuur 9**):

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

Om een beeld in zijn geheel scherper te maken, kun je invoeren:

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

of adaptieve verscherping toepassen:

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

Het omgekeerde kan ook, met het onscherp-masker:

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

Tenslotte proberen we de pixels willekeurig te verstrooien binnen de opgegeven straal:

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

Het resultaat is te zien in **figuur 10**. Speciale effecten
Laten we nu eens bestuderen hoe we speciale effecten op een afbeelding kunnen toepassen. Het eerste effect is *ruis* (noise). Er zijn verschillende soorten ruis. Als eerste gebruiken we Gaussiaanse ruis.

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

De uitvoer is te zien in **figuur 11**. Hieronder geven we een overzicht van alle geldige tekenreeksen die kunnen worden gebruikt als namen van ruistypen:

```
'gaussian'
```

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

We kunnen als volgt blauwverschuiving op een beeld toepassen (**figuur 12**):

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

of een houtskool-effect creëren (**figuur 13**):

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

We kunnen ook een kleurenmatrix toepassen:

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

Een kleurenmatrix kan maximaal 6x6 groot zijn. In een kleurenmatrix staat elke kolom voor een kleurkanaal waarnaar moet worden verwezen, en elke rij voor een kleurkanaal dat moet worden beïnvloed. Voor RGB-afbeeldingen zijn dit rood, groen, blauw, n/a, alpha en een constante (offset). Voor CMYK-afbeeldingen zijn dat cyaan, geel, magenta, zwart, alpha en een constante. In dit voorbeeld hebben we een 3x3-matrix gemaakt met een resultaat als in **figuur 14**.
Andere speciale effecten die door het programma worden ondersteund en in het boek worden beschreven zijn: overvloeien met constante kleur, implosie, Polaroid, sepia, schets, solarisatie, swirl, tint, vignet, golf, wavelet de-noise, flip, flop, en gedraaid.

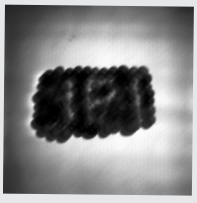
220314-03

Vragen of opmerkingen?

Hebt u vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via ashwin.pajankar@gmail.com of naar de redactie van Elektor via redactie@elektor.com.

Over de auteur

Ashwin Pajankar heeft een Master of Technology van IIIT Hyderabad, en heeft meer dan 25 jaar programmeerervaring. Hij begon zijn carrière in programmeren en elektronica met de programmeertaal BASIC en is nu thuis in assembler, C, C++, Java, Shell Scripting en Python. Zijn verdere technische expertise omvat single-board computers zoals de Raspberry Pi, Banana Pro en Arduino.

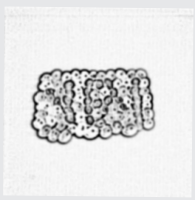


Figuur 9. Reliëf bij een afbeelding in grijstinten.

Figuur 10. Spreidingseffect.

Figuur 11. Gaussiaanse ruis.

Figuur 12. Blauwverschoven afbeelding.



Figuur 13. Houtskool-effect.

Figuur 14. Kleurmatrix-effect.

WEB LINKS

[1] ImageMagick-download:
https://download.imagemagick.org/ImageMagick/download/binaries/ImageMagick-x86_64-apple-darwin20.1.0.tar.gz
[2] ImageMagick voor Win64-systemen:
https://download.imagemagick.org/ImageMagick/download/binaries/ImageMagick-7.1.0-10-Q16-HDRI-x64-dll.exe
[3] ImageMagick voor Win32-systemen:
https://download.imagemagick.org/ImageMagick/download/binaries/ImageMagick-7.1.0-10-Q16-HDRI-x86-dll.exe
[4] Afbeelding-databases, ImageProcessingPlace.com: http://www.imageprocessingplace.com/root_files_V3/image_databases.htm