Convolution d’images

Le script python convolution.py permet d’appliquer les filtres sur une image, une vidéo ou une webcam.

Exemple d’utilisation :

Image
python convolution.py image .\RK_Data\lena.jpg

Video
python convolution.py video '.\RK_Data\test_ video.mp4'

Webcam
python convolution.py webcam

Voici l’image utilisée pour la suite du rapport :

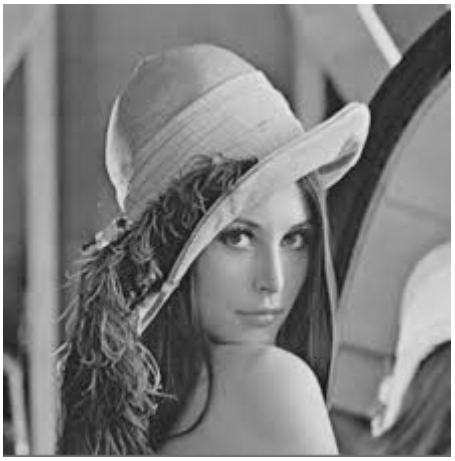
Conversion en niveau de gris

Code :

```
img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

La fonction cv2.cvtColor permet de convertir une image en entrée d’un espace de couleur vers un autre espace de couleur. Dans notre cas, on veut convertir une image en couleur vers une image en niveau de gris, on utilise donc le code : cv2.COLOR_BGR2GRAY.

Résultat :



niveau de gris

Filtre moyenneur

Code :

```
moyenneur_kernel = np.array([[1, 1, 1, 1, 1],
                             [1, 1, 1, 1, 1],
                             [1, 1, 1, 1, 1],
                             [1, 1, 1, 1, 1],
                             [1, 1, 1, 1, 1]]) / 30
cv2.filter2D(src=img, dst=img2, ddepth=-1, kernel=moyenneur_kernel)
```

On initialise le noyau 5*5 puis on applique le filtre via la fonction cv2.filter2D qui permet d'appliquer un filtre à une image. Les deux premiers paramètres sont les images d'entrée et de sortie, ddepth correspond à la profondeur de l'image (-1 signifie que l'on souhaite avoir la même profondeur que l'image d'entrée), le dernier paramètre correspond au noyau que l'on souhaite appliquer.

Résultat :

filtre moyenneur (5*5)

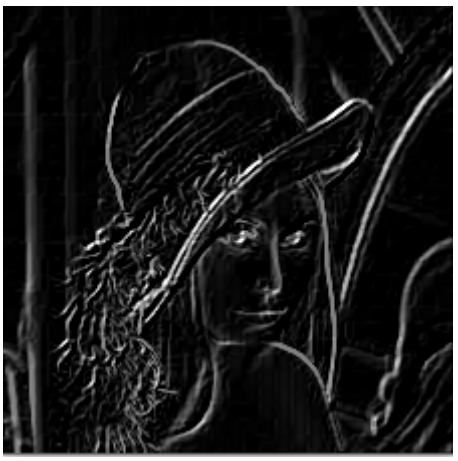
Filtre Sobel

Code :

```
sobel_kernel_x = np.array([[-1, 0, 1],
                           [-2, 0, 2],
                           [-1, 0, 1]])
sobel_kernel_y = np.array([[-1, -2, -1],
                           [0, 0, 0],
                           [1, 2, 1]])
cv2.filter2D(src=img, dst=g_x, ddepth=-1, kernel=sobel_kernel_x)
cv2.filter2D(src=img, dst=g_y, ddepth=-1, kernel=sobel_kernel_y)
cv2.addWeighted(g_x, 0.5, g_y, 0.5, 0, img_sobel)
```

Les noyaux horizontaux et verticaux sont créés avec les 2 premières lignes. Ces noyaux sont ensuite appliqués aux images afin d'avoir les gradients horizontaux et verticaux qui sont fusionnés à la dernière ligne afin de créer l'image de sortie.

Résultat :



filtre Sobel

Autres filtres

Blur :

```
img_blur = cv2.blur(img, (10, 10))
```

La fonction cv2.blur est ici utilisée avec un noyau 10*10. Le résultat obtenu est une image avec un filtre plus important qu'avec la matrice 5*5. Le résultat serait le même si on utilisait le même noyau 5*5 que celui utilisé ci-dessus.

Résultat :

cv2.blur (10*10)

cv2.blur (5*5)

Bilateral filter :

```
img_bil = cv2.bilateralFilter(src=img, d=-1, sigmaColor=75, sigmaSpace=75)
```

Voici le résultat obtenu en appliquant un filtre bilatéral. C'est un filtre qui permet de réduire le bruit dans l'image et d'avoir des bords plus visibles. Les lignes dans le code permettant d'avoir un filtre bilatéral sont mises en commentaire car ce filtre prend beaucoup de temps à l'exécution et ne permet pas d'avoir une vidéo fluide.

Résultat :

Filtre bilatéral