OpenCV

Sources : learnopencv.com, docs.opencv.org

OpenCV est l'une des bibliothèques les plus utilisées pour le traitement d'image.

Avec OpenCV, une image est représentée par l'objet **Mat**, contenant entre autres la **dimension** de l'image, le **nombre de composantes par pixel** (N&B ou couleur), et les **pixels** définissant l'image.

Importer la librairie

```
import cv2
from matplotlib import pyplot as plot
print(cv2.__version__) #remplacer __version__ par la version souhaitée
```

Lire une image

Fonction imread()

Si l'image ne peut pas être lue en raison d'un format de fichier non pris en charge, d'un fichier manquant, d'un format non pris en charge ou invalide, la fonction **imread()** renvoie une matrice.

```
#Lecture du fichier image
img = cv2.imread(filename,[flag])

#On affiche img qu'on renomme 'image'
cv2.imshow('image',img)
```

Paramètres

`filename` : nom du fichier à lire

`flag` : couleur de l'image chargée

- **CV_LOAD_IMAGE_ANYDEPTH -** Si nous le définissons comme un indicateur, il renverra une image de 16 bits / 32 bits lorsque l'entrée a la profondeur correspondante, sinon elle sera convertie en 8 bits.
- **CV_LOAD_IMAGE_COLOR -** Si nous le définissons comme un drapeau, il retourne toujours l'image convertie en couleur.
- **C V_LOAD_IMAGE_GRAYSCALE -** Si nous le définissons comme un drapeau, il convertira toujours l'image en niveaux de gris.

Formats supportés

Window bitmaps *.bmp, *.dib

JPEG files *.jpeg, *.jpg, *.jpe

Portable Network Graphics *.png

Portable image format *.pbm, *.pgm, *.ppm

TIFF files *.tiff, *.tif

Sauvegarder une image

```
cv2.imwrite(filename,img,[params])
```

Paramètres

`filename` - Nom du fichier sauvegardé

`img` - Image à sauvegarder

`params` -

- Pour JPEG, la qualité peut être de 0 à 100. La valeur par défaut est 95.
- Pour PNG, la qualité peut être le niveau de compression de 0 à 9. La valeur par défaut est 1.
- Pour PPM, PGM ou PBM, il peut s'agir d'un indicateur de format binaire 0 ou 1. La valeur par défaut est 1.

Afficher une image

```
cv2.imshow('color image', window_name1, img_color)  #Image en couleur
cv2.imshow('grayscale image', window_name2, img_grayscale) #Image en nuances de gris
cv2.imshow('unchanged image',window_name3, img_unchanged) #Image originale
```

Pour afficher plusieurs images à la fois, spécifiez un nouveau “window_name” pour chaque image que vous souhaitez afficher.

Attendre un clic de l’utilisateur pour exécuter la suite du programme

```
# Attend que l'utilisateur appuie sur une touche
cv2.waitKey(0)
```

La fonction `waitKey()` est une fonction de liaison de clavier. Elle est liée à l'affichage d'une image `imshow()`.

- Elle prend un argument unique, qui est le temps (**en millisecondes**) pendant lequel la fenêtre sera affichée.
- Si l'utilisateur appuie sur une touche pendant cette période, le programme continue.
- Si 0 est passé, le programme attend indéfiniment une frappe de touche.
- Vous pouvez également configurer la fonction pour détecter des frappes de touches spécifiques comme la touche Q ou la touche ESC sur le clavier, ce qui permet de préciser plus explicitement quelle touche déclenchera quel comportement.

Détruire toutes les fenêtres ouvertes

```
# Détruit toutes les fenêtres ouvertes
cv2.destroyAllwindows()
```

La fonction `destroyAllWindows()` détruit toutes les fenêtres créées. Si une fenêtre spécifique doit être détruite, donnez ce nom de fenêtre exact en argument. L'utilisation de `destroyAllWindows()` efface également la fenêtre ou l'image de la mémoire principale du système.

Lire et sauvegarder une vidéo

La lecture et l'écriture de vidéos dans OpenCV sont très similaires à la lecture et l'écriture d'images. Une vidéo n'est rien d'autre qu'une série d'images appelées *frames*. Tout ce que vous avez à faire est de boucler sur tous les *frames* d'une séquence vidéo, puis de traiter un *frame* à la fois.

Ouvrir un fichier vidéo

`cv2.VideoCapture(path , apiPreference)` crée un objet de capture vidéo.

Il faut toujours créer une boucle `if` afin de vérifier que la vidéo est bien ouverte.

```
import cv2

# On ouvre un fichier vidéo et on le stock dans l'objet "vid_capture"
vid_capture = cv2.VideoCapture('Resources/Cars.mp4')

if (vid_capture.isOpened() == False):
  print("Une erreur survient lors de l'ouverture de la vidéo")
else:
 #Programme à exécuter
```

Pour en savoir plus sur les API :

apiPreference

Obtenir le FPS

```
if (vid_capture.isOpened() == False):
  print("Error opening the video file")
else:
  # Get frame rate information

  fps = int(vid_capture.get(5))
  print("Frame Rate : ",fps,"frames per second")

  # Get frame count
  frame_count = vid_capture.get(7)
  print("Frame count : ", frame_count)
```

Obtenir la taille de la vidéo

```
# Obtain frame size information using get() method
frame_width = int(vid_capture.get(3))
frame_height = int(vid_capture.get(4))
frame_size = (frame_width,frame_height)
fps = 20
```

Lire une vidéo provenant d’une Webcam

Plutôt que de spécifier un emplacement source pour un fichier vidéo ou une séquence d'images, il vous suffit simplement de donner un index de périphérique de capture vidéo, comme indiqué ci-dessous :

- Si votre système dispose d'une webcam intégrée, l'index de périphérique de la caméra sera « `0` ».
- Si vous avez plus d'une caméra connectée à votre système, l'index de périphérique associé à chaque caméra supplémentaire est incrémenté (par exemple, `1`, `2`, etc).

```
vid_capture = cv2.VideoCapture(0, "apiPreference")
```

Sauvegarder une vidéo

La classe `VideoWriter()` prend les arguments suivants :

- `filename` : chemin d'accès du fichier vidéo de sortie
- `apiPreference` : identifiant des API
- `fourcc` : code à 4 caractères du codec utilisé pour compresser les images

> Utilisez les spécifications *fourcc* suivantes pour créer des formats AVI ou MP4 : AVI: `cv2.VideoWriter_fourcc('M','J','P','G')` MP4: `cv2.VideoWriter_fourcc(*'XVID')`

- `fps` : fréquence d'images du flux vidéo créé
- `frame_size` : taille des images vidéo
- `isColor` : si différent de zéro, l'encodeur attendra et encodera des images couleur; sinon, il fonctionnera avec des images en niveaux de gris

```
while(vid_capture.isOpened()):
    # vid_capture.read() methods returns a tuple, first element is a bool
    # and the second is frame

    ret, frame = vid_capture.read()
    if ret == True:
           # Write the frame to the output files
           output.write(frame)
    else:
```

```
            print('Stream disconnected')
              break
```

Attention :

À cette étape, plusieurs erreurs peuvent se produire. Les plus courantes sont l'**erreur de taille de trame** et l'**erreur de préférence API**. Si la taille de la trame n'est pas similaire à celle de la vidéo, même si nous obtenons un fichier vidéo dans le répertoire de sortie, il sera vide. Si vous utilisez la méthode de forme NumPy pour récupérer la taille de la trame, n'oubliez pas d'inverser la sortie car OpenCV renverra ***hauteur x largeur x canaux***. S'il génère une erreur de préférence d'API, nous devrons peut-être passer le drapeau `CAP_ANY` dans l'argument `VideoCapture()`. Cela peut être vu dans l'exemple de webcam, où nous utilisons `CAP_DSHOW` pour éviter les avertissements qui se génèrent.

Libérer la mémoire

```
vid_capture.release()
output.release()
```

Redimensionner une image

- La **réduction** de la taille d'une image nécessitera un rééchantillonnage des pixels.
- **Augmenter** la taille d'une image nécessite la reconstruction de l'image. Cela signifie que vous devez interpoler de nouveaux pixels.

Obtenir les dimensions de l'image

Deux possibilités :

`image.shape` retourne trois valeurs : hauteur, largeur et nombre de canaux

La fonction `size()`

- `image.size().width` retourne la largeur

- `image.size().height` retourne la hauteur

Fonction `resize()`

Syntaxe : `resize(src, dsize, dst, fx, fy, interpolation)`

- `src`: image source
- `dsize`: taille souhaitée de l'image de sortie; liste de taille 2 : L =(largeur,hauteur)
- `fx`: facteur d'échelle le long de l'axe horizontal
- `fy`: facteur d'échelle le long de l'axe vertical

  Si fx=fy, cela permet de garder le même ratio
- `interpolation`: donne la possibilité de différentes méthodes de redimensionnement de l'image

▼ Les différentes méthodes d'interpolation

- `INTER_AREA`: `INTER_AREA` utilise la relation de zone de pixel pour le rééchantillonnage. Cela convient le mieux pour réduire la taille d'une image (rétrécissement). Lorsqu'il est utilisé pour zoomer sur l'image, il utilise la méthode `INTER_NEAREST`.
- * `INTER_CUBIC`: Ceci utilise une interpolation bicubique pour redimensionner l'image. Pendant le redimensionnement et l'interpolation des nouveaux pixels, cette méthode agit sur les 4 × 4 pixels voisins de l'image. Il prend ensuite la moyenne pondérée des 16 pixels pour créer le nouveau pixel interpolé.
- * `INTER_LINEAR` : Cette méthode est quelque peu similaire à l'interpolation `INTER_CUBIC`. Mais contrairement à `INTER_CUBIC`, cela utilise 2x2 pixels voisins pour obtenir la moyenne pondérée pour le pixel interpolé.
- * `INTER_NEAREST`: La méthode `INTER_NEAREST` utilise le concept du voisin le plus proche pour l'interpolation. C'est l'une des méthodes les plus simples, n'utilisant qu'un seul pixel voisin de l'image pour l'interpolation.

Découper une image

L'image découpée est `img(Range(start_row, end_row), Range(start_col, end_col))`

- La première dimension est toujours le nombre de lignes ou la hauteur de l'image.
- La deuxième dimension est le nombre de colonnes ou la largeur de l'image.

```
# Importation des librairies
import cv2
import numpy as np

img = cv2.imread('test.jpg')
print(img.shape) # On affiche les dimensions de l'image originale
cv2.imshow("original", img)

# Découpage de l'image
cropped_image = img[80:280, 150:330]

# DAffichage de l'image découpée
cv2.imshow("cropped", cropped_image)

# Sauvegarder l'image découpée
cv2.imwrite("Cropped Image.jpg", cropped_image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Pour découper une image en plusieurs pièces, on peut utiliser le code suivant :

```
#On récupère les dimensions de l'image initiale
img =  cv2.imread("test_cropped.jpg")
image_copy = img.copy()
imgheight=img.shape[0] #hauteur de l'image
imgwidth=img.shape[1] #largeur de l'image

#On définit la taille des pièces à découper
M = 76 #Hauteur de la pièce
N = 104 #Largeur de la pièce

x1 = 0
y1 = 0

#On parcourt pièce par pièce l'image
for y in range(0, imgheight, M):
    for x in range(0, imgwidth, N):
        if (imgheight - y) < M or (imgwidth - x) < N:
            break
```

```
        y1 = y + M
        x1 = x + N

        # On vérifie que les dimensions de la pièce n'excèdent pas celles de l'image
        if x1 >= imgwidth and y1 >= imgheight:
            x1 = imgwidth - 1
            y1 = imgheight - 1
            #Découpage
            tiles = image_copy[y:y+M, x:x+N]
            #Sauvegarde de chaque pièce
            cv2.imwrite('saved_patches/'+'tile'+str(x)+'_'+str(y)+'.jpg', tiles)
            cv2.rectangle(img, (x, y), (x1, y1), (0, 255, 0), 1)

        elif y1 >= imgheight: # Si la hauteur de la pièce excède celle de l'image
            y1 = imgheight - 1
            #Découpage
            tiles = image_copy[y:y+M, x:x+N]
            #Sauvegarde
            cv2.imwrite('saved_patches/'+'tile'+str(x)+'_'+str(y)+'.jpg', tiles)
            cv2.rectangle(img, (x, y), (x1, y1), (0, 255, 0), 1)

        elif x1 >= imgwidth: # Si la largeur de la pièce excède celle de l'image
            x1 = imgwidth - 1
            #Découpage
            tiles = image_copy[y:y+M, x:x+N]
            #Sauvegarde
            cv2.imwrite('saved_patches/'+'tile'+str(x)+'_'+str(y)+'.jpg', tiles)
            cv2.rectangle(img, (x, y), (x1, y1), (0, 255, 0), 1)

          else: #Si aucune des deux dimensions ne convient
            #Découpage
            tiles = image_copy[y:y+M, x:x+N]
            #Sauvegarde
            cv2.imwrite('saved_patches/'+'tile'+str(x)+'_'+str(y)+'.jpg', tiles)
            cv2.rectangle(img, (x, y), (x1, y1), (0, 255, 0), 1)
```

Rotation de l’image

→ On utilise la fonction `getRotationMatrix2D(centre, angle, échelle)` dont la construction se base sur la matrice rotation connue.

La fonction `getRotationMatrix2D()` prend les arguments suivants :

- `centre` : le centre de rotation de l'image d'entrée
- `angle` : l'angle de rotation en degrés
- `échelle` : un facteur d'échelle isotropique qui agrandit ou réduit l'image en fonction de la valeur fournie

Si l'angle est positif, l'image est tournée dans le sens contraire des aiguilles d'une montre. Si vous voulez faire tourner l'image dans le sens des aiguilles d'une montre du même montant, alors l'angle doit être négatif.

$$R'_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

→ La fonction `warpAffine()` applique une transformation affine à l'image. Après l'application de cette transformation, toutes les lignes parallèles de l'image d'origine resteront également parallèles dans l'image de sortie.

Syntaxe : `warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])`

Les arguments de la fonction sont les suivants:

- `src`: l'image source
- `M`: la matrice de transformation
- `dsize`: taille de l'image de sortie
- `dst`: l'image de sortie
- `flags`: combinaison de méthodes d'interpolation telles que INTER_LINEAR ou INTER_NEAREST
- `borderMode`: la méthode d'extrapolation des pixels
- `borderValue`: la valeur à utiliser en cas de bordure constante, a une valeur par défaut de 0

```
import cv2

# Importation de l'image
image = cv2.imread('image.jpg')

# On récupère la taille de l'image
height, width = image.shape[:2]
# On obtient les coordonnées du centre de l'image
center = (width/2, height/2)

# Matrice de rotation
rotate_matrix = cv2.getRotationMatrix2D(center=center, angle=45, scale=1)

# Rotation de l'image
```

```
rotated_image = cv2.warpAffine(src=image, M=rotate_matrix, dsize=(width, height))

#Affiche les images obtenues
cv2.imshow('Original image', image)
cv2.imshow('Rotated image', rotated_image)
# Appuie sur n'importe quelle touche du clavier pour fermer la fenêtre
cv2.waitKey(0)
```

Rq : pour translater une image, on procède de la même manière en utilisant la matrice translation.

Dessiner sur l’image

Tracer une ligne

Avant d'appeler la fonction `line()`, créez une copie de l'image d'origine en utilisant la fonction `copy()` . Une copie garantira que toutes les modifications apportées à l'image n'affecteront pas l'image d'origine.

Syntaxe : `line(image, start_point, end_point, color, thickness, lineType)`

- Le premier argument est l'image.
- Les deux arguments suivants sont le point de départ et le point d'arrivée pour la ligne.

▼ lineType

  - LINE_4 : pointillés épais
  - LINE_8 : pointillés fins
  - LINE_AA : anti-aliasing

```
#Copie de l'image originale
imageLine = img.copy()

# Tracer la ligner
pointA = (200,80)
pointB = (450,80)
cv2.line(imageLine, pointA, pointB, (255, 255, 0), thickness=3))

#Affichage
cv2.imshow('Image Line', imageLine)
cv2.waitKey(0)
```

Tracer un cercle

`circle(image, center_coordinates, radius, color, thickness)`

- Comme pour toutes les fonctions de dessin dans OpenCV, le premier argument est l'image.
- Les deux arguments suivants définissent les coordonnées du centre du cercle et de son rayon.
- Les deux derniers arguments spécifient la couleur et l'épaisseur de la ligne.

→ Pour remplir le cercle, mettre en argument `thickness=-1`.

Tracer un rectangle

`rectangle(image, start_point, end_point, color, thickness)`

Dans la fonction `rectangle()`, vous fournissez le point de départ (en haut à gauche) et le point d'arrivée (en bas à droite) pour les coins du rectangle.

Tracer une ellipse

`ellipse(image, centerCoordinates, axesLength, angle, startAngle, endAngle, color, thickness)`

La syntaxe de la fonction `ellipse()` est assez similaire à celle du cercle. Sauf qu'à la place du rayon, il faut spécifier les éléments suivants :

- les longueurs majeures et mineures des axes de l'ellipse
- l'angle de rotation
- l'angle de départ et d'arrivée de l'ellipse

Ajouter du texte sur l'image

`putText(image, text, org, font, fontScale, color)`

- `image` : l'image d'entrée.
- `text` : chaîne de texte réelle avec laquelle nous voulons annoter l'image.
- `org` : emplacement de départ pour le coin supérieur gauche de la chaîne de texte.
- ▼ `font` : style de police

  FONT_HERSHEY_SIMPLEX = 0,

  FONT_HERSHEY_PLAIN = 1,

FONT_HERSHEY_DUPLEX = 2,

FONT_HERSHEY_COMPLEX = 3,

FONT_HERSHEY_TRIPLEX = 4,

FONT_HERSHEY_COMPLEX_SMALL = 5,

FONT_HERSHEY_SCRIPT_SIMPLEX = 6,

FONT_HERSHEY_SCRIPT_COMPLEX = 7,

FONT_ITALIC = 16

- `fontScale` : échelle de police; valeur en virgule flottante, utilisée pour augmenter ou diminuer la taille de base de la police.
- `color` : spécifiée comme un triplet BGR.

Opérations simples

Accéder à un pixel de l’image

```
import numpy as np
import cv2
img = cv2.imread("C:\Users\Image.jpeg")
pixel = img[100,100]
print(pixel)
```
