2

Écran de contrôle de caméra vidéo – Gestion des processus légers

Dans ce chapitre, nous allons écrire un programme permettant d'effectuer l'acquisition d'images à partir de plusieurs caméras.

Nous procéderons par étapes : dans un premier exemple, nous présenterons les principes de bases de l'acquisition vidéo, puis dans le deuxième, nous verrons la gestion de la ligne de commande, la gestion d'un fichier de configuration, la gestion de la souris, la modification de la résolution de l'acquisition vidéo et le calcul sur le flux seront ajoutés. Enfin, dans le exemple final, l'utilisateur pourra choisir pendant l'exécution du programme d'appliquer un filtre médian à l'image. L'acquisition de l'image et l'application du filtre médian seront gérées par un processus léger (ou *thread*) et l'affichage des images par le programme principal. L'affichage pourra se faire dans une fenêtre pour chaque caméra ou bien dans une fenêtre unique pour toutes les caméras. Dans ce dernier cas, un clic dans l'image ouvrira une nouvelle fenêtre ne contenant que les images issues de la caméra sélectionnée. Le zoom pourra aussi être choisi. Ces paramètres seront fixés au démarrage du programme en utilisant les arguments de la ligne de commande.

Attention *› Le système d'exploitation peut bloquer l'accès aux caméras. Si le programme ne trouve aucune des caméras reliées à votre ordinateur, vérifiez les paramètres de sécurité de votre système.*

2.1. Acquisition de plusieurs flux vidéo

Exemple 2.1 : *Code source (`ch2ex1`) dans le dossier `py_threads`*

Dans le programme Python suivant, nous effectuons l'acquisition des images issues des flux vidéo reliés à l'ordinateur. Chaque image est affichée dans une fenêtre. Le programme s'arrête lorsque l'utilisateur appuie sur la touche Echap du clavier.

Importation du module OpenCV

Pour écrire un programme Python basé sur OpenCV, il faut importer les modules numpy ❶ et cv2 ❷.

```
import numpy as np ❶
import cv2 as cv ❷
```

L'alias du module cv2 est cv ❷. Pourquoi cet alias ? Le fait est que toutes les fonctions de la documentation officielle utilisent l'alias cv.

Les images dans OpenCV sont transcrites en type numpy array (voir en fin d'ouvrage l'annexe Memento d'OpenCV).


Afficher le listing suivant dans un navigateur

Le programme importe les librairies numpy ❶, cv2 ❷ et time ❸. Le module time est utilisé pour estimer le temps d'acquisition et d'affichage par seconde d'une image.

Une liste vide est créée pour contenir les flux matériels (les webcams) reliés à votre ordinateur ❹. Les flux matériels sont indexés à partir de 0 (entre ❹ et ❺). La boucle ❺ ouvre le périphérique de capture indexé i en créant un objet cv.VideoCapture ❻. Si le périphérique est ouvert, la méthode cv.VideoCapture.isOpened ❼ de la classe cv.VideoCapture retournera la valeur vraie.

Classe *cv.VideoCapture*

La classe cv.VideoCapture permet d'accéder aux caméras reliées à l'ordinateur en utilisant l'interface de programmation (API). Les interfaces connues par OpenCV sont précisées dans la documentation. Si vous avez téléchargé OpenCV, il vous faut utiliser la commande cv.getBuildInformation() pour connaître les détails de votre version Python d'OpenCV :

```
print(cv.getBuildInformation())
```

En particulier, les parties GUI (Graphics Unit Interface), Media I/O et Video I/O vous renseignent sur les API accessibles dans votre version d'OpenCV.

```
  Video I/O:
    DC1394:                      NO
    FFMPEG:                      YES (prebuilt binaries)
      avcodec:                   YES (58.35.100)
      avformat:                  YES (58.20.100)
      avutil:                    YES (56.22.100)
      swscale:                   YES (5.3.100)
      avresample:                YES (4.0.0)
    GStreamer:                   NO
    DirectShow:                  YES
    Media Foundation:            YES
      DXVA:                      YES
```

Par exemple, si vous souhaitez ouvrir une caméra accessible à l'aide de l'API V4L/V4L2, vous devez instancier un objet `cv.VideoCapture` avec `i+cv.CAP_4L` en paramètre : `cv.VideoCapture(i+cv.CAP_4L)`.

Si l'interface vous est indifférente, alors utilisez simplement `cv.VideoCapture(i)`.

Si le périphérique est ouvert, on l'ajoute à la liste `liste_webcam` (entre ❼ et ❽) sinon on quitte la boucle en ❽. Si la liste `liste_webcam` est vide ❾, aucun périphérique n'est présent ; on écrit un message et on quitte le programme. Entre ❿ et ⓫, on initialise la variable `code_touche_clavier` utilisée pour lire une touche de clavier, la variable `nb_prises` pour compter le nombre d'acquisition et la variable `tps_ini` pour initialiser le chronomètre.

```
import time ❶
import numpy as np ❷
import cv2 as cv ❸

CODE_TOUCHE_ECHAP = 27
liste_webcam = [] ❹
index_camera_ouvertes = 0
while True: ❺
    webcam = cv.VideoCapture(index_camera_ouvertes) ❻
    if webcam.isOpened(): ❼
        liste_webcam.append(webcam)
    else:
        break ❽
    index_camera_ouvertes += 1
if not liste_webcam: ❾
    print("Aucune camera reliee")
    exit()

code_touche_clavier = 0 ❿
```

```
nb_prises = 0
tps_ini = time.clock() ⓫

while True: ⓬
    for idx, cam in enumerate(liste_webcam): ⓭
        ret, img = cam.read() ⓮
        if ret: ⓯
            cv.imshow('webcam b' + str(idx), img) ⓰
        else:
            print(" image ne peut être lue")
    nb_prises += 1 ⓱
    if nb_prises == 100: ⓲
        nb_images_par_seconde = nb_prises / (time.clock() - tps_ini)
        print("Pour chaque camera : ", nb_images_par_seconde, " Images
 par seconde")
        tps_ini = time.time()
        nb_prises = 0
    code_touche_clavier = cv.waitKey(20) ⓳
    if code_touche_clavier == CODE_TOUCHE_ECHAP: ⓴
        break

for cam in liste_webcam: ㉑
    cam.release() ㉒
cv.destroyAllWindows() ㉓
```

La boucle d'acquisition et d'affichage des images est entre ⓬ et ⓴. En ⓭, on parcourt la liste des flux ouverts, puis on appelle la méthode `read` ⓮ de classe `cv.VideoCapture`. Les valeurs de retour sont un booléen et l'image lue. La valeur du booléen est vraie lorsque la lecture est valide.

Si la valeur de retour est vraie ⓯, une image est alors disponible et on peut l'afficher en appelant la fonction `cv.imshow` ⓰ avec en premier argument le nom de la fenêtre et en second l'objet `array` contenant l'image. En ⓱, le nombre d'acquisitions est incrémenté et si ce nombre est de 100 ⓲, on affiche le nombre d'images capturées par seconde (entre ⓲ et ⓳). L'appel à `cv.waitKey` permet de lire le code de la touche clavier pressée par l'utilisateur et d'afficher les images dans toutes les fenêtres. Le seul paramètre de cette fonction est le délai minimum en milliseconde pour attendre l'appui d'une touche sur le clavier. La valeur de retour est −1 si aucune touche n'a été pressée. Si le code de la touche est égal à 27 ⓴ (touche Echap du clavier), on quittera la boucle.

Afficher une image

La fonction `cv.imshow` affiche une image (un objet `array`). Il faut bien noter que l'affichage est fait lors de l'appel à la fonction `cv.waitKey`. La fonction `cv.imshow` en-