École Centrale

TP 2 - Calibration de caméras

OpenCV est une bibliothèque Open Source de vision par ordinateur très utilisée dans les domaines académiques et même dans l'industrie. Elle permet en effet d'utiliser de nombreux méthodes et algorithmes à la pointe dans les domaines tels que :

- le traitement d'images ;
- l'analyse vidéo ;
- la détection d'objets et/ou de points d'intérêts ;
- etc

OpenCV est une bibliothèque découpée en plusieurs modules, dans la séance d'aujourd'hui, nous allons utiliser les modules suivants :

- `core` : le module de base d'OpenCV contenant les classes de base ;
- `highgui` : module contenant les classes des interfaces utilisateurs (fenêtre, gestion clavier et souris, etc.) ;
- `imgproc` : module de traitement d'image (par exemple conversion d'une image couleur vers niveaux de gris, etc.) ;
- `calib3d` : module de calibration 3D et stéréoscopique (détection d'un échiquier, etc.).

Tout au long du TP merci de vous référer à la documentation OpenCV `https://docs.opencv.org/4.x/`

1 Prise en main - lecture vidéo et fichiers images

Dans cette partie, nous allons vous familiariser avec l'utilisation d'OpenCV. Pour ce faire, nous allons vous faire écrire une application qui va :

- afficher les images issues d'une caméra ou de différents fichiers en boucle jusqu'à ce que l'utilisateur appuie sur la touche 'esc' ;

— convertir une image couleur en niveaux de gris ;
— offrir la possibilité à l'utilisateur de choisir entre l'affichage de l'image en couleur ou l'image en niveaux de gris ;
— libérer proprement les ressources avant de quitter votre programme.

1.1 Squelette de l'application

A partir des exemples du TP2, créez un fichier OpenCV-calibration-part1.py.
Commencez par inclure les entêtes OpenCV que nous allons utiliser. Vous aurez également besoin de définir une constante représentant le code ASCII de la touche 'ESC' et espace du clavier, que nous voulons pouvoir reconnaitre pour quitter l'application et une méthode principale.

```
# openCV import
import cv2 as cv

# Keycode definitions
ESC_KEY = 27
Q_KEY = 113

def main():
    # Define variables

    # A key that we use to store the user keyboard input
    key = None

    # Starting the code

if __name__ == "__main__":
    main()
```

1.2 Affichage du flux vidéo

Dans un premier temps, nous voulons pouvoir récupérer le flux d'une caméra vidéo (assurez-vous d'en avoir une connectée à votre ordinateur).
Pour ce faire, nous allons utiliser `VideoCapture` du module `cv2` d'OpenCV.
Un objet de type `VideoCapture` peut ouvrir le flux d'une caméra connectée à l'ordinateur, ou bien ouvrir un fichier vidéo. Nous allons l'utiliser dans le premier cas de figure ici, mais vous pourrez essayer par vous-même d'ouvrir un fichier vidéo.
Pour ce faire, nous allons utiliser la méthode `open` dont la documentation est ici : `https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html`

Cette méthode prend en paramètre soit :
— une chaine de caractères contenant le nom du fichier vidéo que l'on souhaite ouvrir ;

— un entier représentant l'identifiant du dispositif (caméra) que l'on souhaite ouvrir (en schématisant : indice de la ou des caméra(s) connectée(s), commençant à 0).

Vérifiez ensuite que la capture a pu être effectuée. Pour ce faire, utilisez la méthode `cap.isOpened()` retournant un booléen permettant de savoir si tout s'est bien passé.

Modifiez votre programme pour demander en boucle à l'utilisateur un numéro de caméra tant que la capture n'est pas ouverte (ou -1 pour arrêter complètement l'application).

Avant toute chose, nous souhaitons manipuler des images issues de la caméra. Pour ce faire, déclarez deux objets (qui seront de type `Mat` de manière implicite) qui permettent de stocker des images (couleur, puis niveau de gris) en OpenCV.
Nous allons aussi avoir besoin d'une fenêtre pour afficher les résultats, pour ce faire, nous allons utiliser la méthode `namedWindow` du module `cv2`, qui permet de créer une fenêtre, laquelle est identifiée par une chaîne de caractères représentant son titre :

```
# Creating a window to display the images
windowName = "OpenCV Calibration"
cv2.namedWindow(windowName, WINDOW_AUTOSIZE)
```

La fenêtre ne doit être créée qu'une seule fois !

Pour extraire une image de la capture, il suffit d'utiliser la méthode `read` de la classe `VideoCapture` :

```
 # Capture frame-by-frame
 # if image is read correctly ret is True
ret, image = cap.read()
```

Pour afficher une image dans une fenêtre préalablement créée, on utilisera la méthode `imshow`

du module `cv2`. Cette méthode prend en paramètre le nom de la fenêtre dans laquelle on veut afficher, et l'image que l'on souhaite afficher. Par exemple, (on supposera que image contient bel et bien une image) :

```
# Showing the image in the window
cv2.imshow(windowName,image);
```

La dernière chose que vous devez mettre en œuvre est d'encapsuler la récupération d'une image, puis son affichage dans une boucle. Cette dernière doit s'arrêter lorsque l'utilisateur appuie sur la touche 'ESC' de son clavier. Pour pouvoir récupérer les entrées clavier de l'utilisateur, nous allons utiliser une nouvelle méthode du module `cv2` : `waitKey`. Cette méthode prend comme argument un entier qui représente le délai pendant lequel la méthode va attendre un évènement utilisateur. Si cet entier est négatif ou nul, alors la méthode est bloquante, et `waitKey` attend indéfiniment pour l'évènement.
Enfin, afin de libérer les ressources que vous avez allouées pendant la création de cette application, n'oubliez pas d'appeler les méthodes suivantes, pour libérer successivement les fenêtres que vous avez créées, et la capture vidéo :

```
    # Destroying the windows
    cv2.destroyWindow(window_name)
    cv2.destroyAllWindows()

    # Releasing the video capture
    cap.release();
```

Vous devriez maintenant voir le flux vidéo de votre webcam s'afficher dans la fenêtre OpenCV créée.

Nous allons rajouter une dernière étape à notre affichage de flux vidéo : la possibilité de transformer notre image en niveaux de gris, et de choisir si l'on souhaite afficher l'image en couleurs ou en niveaux de gris.

La conversion d'une image de couleur en niveaux de gris est très simple en OpenCV, il suffit d'utiliser la méthode `cvtColor` du module `cv2`. Cette méthode prend en entrée l'image à convertir, l'image résultante et un paramètre déterminant les différentes options de conversion (voir la documentation pour plus de détails).

Ajoutez dans votre code la ligne qui permet de convertir la matrice `image` en une matrice `gray_image` en niveaux de gris.

Enfin, nous voulons que lorsque l'utilisateur appuie sur la touche 'g' alors on affiche l'image en niveaux de gris. Si l'on appuie une nouvelle fois sur 'g', alors on réaffiche l'image en couleurs. Notez que la fonction `waitKey` retourne un caractère.

Vous rendrez le code de cette première partie et ajouterez des captures d'écrans dans votre compte rendu pour montrer le bon fonctionnement de votre code.

1.3 Affichage de photos

Dans cette deuxième partie de prise en main, nous allons modifier notre programme afin d'afficher des fichiers image à la place d'un flux vidéo. Pour ce faire, vous utiliserez les images fournies sur le serveur pédagogique dans le dossier "calib_gopro". Créez un nouveau `main` à partir du travail de la partie précédente.

Pour lire un fichier image, nous allons utiliser la méthode `imread` du module cv2 en python.

Modifiez votre méthode `main` afin que votre boucle d'affichage du flux vidéo de la caméra, affiche en boucle les images contenues dans le dossier "calib_gopro". Ces images sont toutes nommées GOPR84, suivi d'un numéro (précédé de 0 pour les numéros inférieurs à 10) puis possèdent l'extension .JPG. Il existe 27 images allant de GOPR8401.JPG à GOPR8427.JPG.

Pour cette partie, il se peut que vous deviez utiliser une méthode de conversion d'un nombre entier en chaine de caractères.

En python, cela est faisable via la fonction `str` :

```
msg = "bli"
monEntier = 12
msg += str(monEntier)
```

Vous rendrez également le code de cette deuxième partie.

Dans la suite du TP, afin d'illustrer l'intérêt de la calibration de caméra, vous devrez utiliser les photos du dossier "calib_gopro", et les comparer avec les résultats que vous obtiendrez sur les autres fichiers d'exemples présents sur le serveur pédagogique.