TP3 Calibration de caméras

(En Python)

1 Prise en main - lecture vidéo et fichiers images

1.1 Squelette de l'application

Dans cette partie, nous définissons le cadre principal du programme. Déclarez d'abord les bibliothèques Python nécessaires, définissez le code ASCII correspondant à ESC et à la barre d'espace, et enfin définissez main (), qui est la méthode principale du programme.

```
import cv2, os
import os.path as osp
import numpy as np
import glob

ESC_KEY = 27
Q_KEY = 113

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

1.2/1.3 Affichage du flux vidéo

Nous avons d'abord défini une méthode pour ouvrir la vidéo ou la caméra.

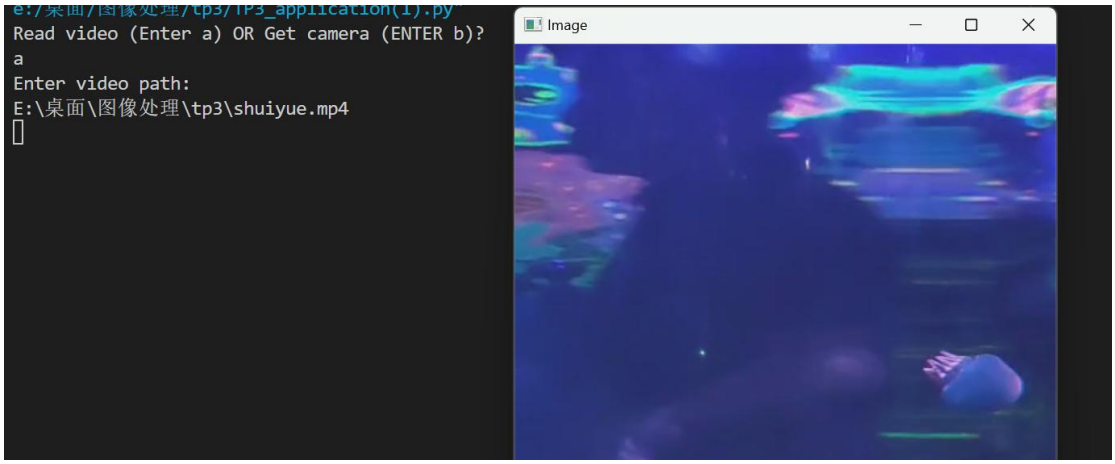
Lors de l'exécution de la méthode, nous sommes invités à obtenir l'entrée d'un utilisateur. Si l'utilisateur tape "a", il est invité à entrer le chemin du vidéo, puis à afficher la vidéo en boucle. Si l'utilisateur entre "b", l'utilisateur est invité à entrer le numéro de la caméra. S'il ne peut pas allumer la caméra, demander à plusieurs reprises jusqu'à ce que l'utilisateur a entré le bon numéro, ou jusqu'à ce que l'utilisateur a entré "-1" pour quitter le programme.

```
    def display_camera_or_video(self):
        mode = input("Read video (Enter a) OR Get camera (ENTER b)?\n")
        if mode == 'a':
            video_path = input("Enter video path:\n")
            cap = cv2.VideoCapture(video_path)
            if not cap.isOpened():
                print(f"Cannot open {video_path}")
                exit()
            else:
```

```
                frames = []
                while True:
                    ret, frame = cap.read()
                    if not ret:
                        break
                    frames.append(frame)
                return self.display_images_in_loop(frames)

        elif mode == 'b':
            while True:
                camera_id = input("L'identifiant de la caméra utilisée:\n")
                if camera_id == '-1':
                    exit()
                try:
                    camera_id = int(camera_id)
                except ValueError:
                    print("Veuillez entrer un nombre entier!")
                    continue
                cap = cv2.VideoCapture(camera_id)
                if not cap.isOpened():
                    print("ID de caméra incorrect")
                    continue
                else:
                    print("Successfully open your camera!")
                    return self.display_camera()
        else:
            print("Please Press a or b!")
```

Sélectionnez d'ouvrir la vidéo:



Sélectionnez d'ouvrir le caméra:

```
Read video (Enter a) OR Get camera (ENTER b)?
b
Please enter your camera number:
2
[ERROR:0@12.280] global obsensor_uvc_stream_channel.cpp:159 cv::obsensor::getStreamChannelGrou
p Camera index out of range
Cannot open camera
Please enter your camera number:
3
[ERROR:0@14.948] global obsensor_uvc_stream_channel.cpp:159 cv::obsensor::getStreamChannelGrou
p Camera index out of range
Cannot open camera
Please enter your camera number:
1
[ERROR:0@17.360] global obsensor_uvc_stream_channel.cpp:159 cv::obsensor::getStreamChannelGrou
p Camera index out of range
Cannot open camera
Please enter your camera number:
0
Successfully open your camera!
```

Après avoir ouvert la caméra avec succès, les fonctions suivantes peuvent être réalisées:

1. Appuyez sur esc ou q pour quitter;

2. Appuyez sur le bouton g pour que l'écran commute entre l'image en échelle de gris et l'image en couleur;

3. Appuyez sur le bouton S pour enregistrer le cadre actuel;

4. Appuyez sur la touche f pour entrer ou sortir de la détection de grille d'échiquier, si la grille ne peut pas être détectée dans l'écran, il sera suggéré sous l'écran, et les points d'angle seront dessinés si la grille est détectée;

5. Le calibrage de l'objectif sera automatiquement effectué lorsque vous enregistrez suffisamment de n images, et une nouvelle fenêtre apparaîtra pour montrer l'image calibrée.

```
def display_camera(self):
        """
        Help:
        - Press "g" to convert gray and color images
        - Press "s" to save the current frame
        - Press "f" to start and exit chessboard detection
        - Press "q" or "esc" to exit
        - After saving “num_img” images, calibration will automatically begin, and a
    new display window will show the camera image with the distortion calibrated.
        """
        cap = cv2.VideoCapture(self.camera_id)
        grayscale_mode = False
        detect_chessboard_mode = False
        calibrate_camera_mode = False
        save_press = 0
```

```
        while True:
            # Capture frame-by-frame
            ret, frame = cap.read()
            # if frame is read correctly ret is True
            if not ret:
                print("Can't receive frame (stream end?). Exiting ...")
                break
            if grayscale_mode:
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                cv2.imshow('Camera', frame)
            else:
                cv2.imshow('Camera', frame)

            if detect_chessboard_mode:
                frame, _, _ = self.find_corners(frame)
                cv2.imshow('Camera', frame)
            # else:
            #       cv2.imshow('Camera', frame)

            if save_press >= self.num_img:
                save_press = 0 # We can collecte images again.
                print("Image acquisition completed. Starting Camera Calibration!")
                img_frames = read_images_from_folder(self.save_dir)
                ret, mtx, dist, rvecs, tvecs = self.calibrate_camera(img_frames)
                h, w = frame.shape[:2]
                newcameramtx, roi = cv2.getOptimalNewCameraMatrix(mtx, dist,
(w,h), 1, (w,h))
                print("The rest of the image has been calibrated for distortion.")
                calibrate_camera_mode = True # In order not to repeat the print.

            if calibrate_camera_mode:
                frame = cv2.undistort(frame, mtx, dist, None, newcameramtx)
                cv2.imshow('Calibrated Camera', frame)
            else:
                cv2.imshow('Camera', frame)

            key = cv2.waitKey(1)
            if key == ord('g'):
                grayscale_mode = not grayscale_mode
            if key == ESC_KEY or key == Q_KEY:
                break
            if key == ord('s') and save_press < self.num_img:
                cv2.imwrite(osp.join(self.save_dir, str(save_press) +'.png'), frame)
```

```
                save_press += 1
            if key == ord('f'):
                detect_chessboard_mode = not detect_chessboard_mode

        # When everything done, release the capture
        cap.release()
        cv2.destroyAllWindows()
```

1.4 Affichage de photos

Obtenez les images dans un dossier et triez-les:

```
def read_images_from_folder(frame_folder):
    frame_names = sorted([f for f in os.listdir(frame_folder)
                          if f.endswith(('.jpg', '.jpeg', '.png', '.JPG', '.PNG'))])
    frames = [cv2.imread(osp.join(frame_folder, f)) for f in frame_names]
    return frames
```

Cycle afficher les images:

```
def display_images_in_loop(self, frames):
        """
        Help:
        - Press "g" to convert gray and color images
        - Press "s" to save the current frame
        - Press "q" or "esc" to exit
        """
        current_index = 0
        save_press = 0
        grayscale_mode = False

        while True:
            image = frames[current_index]

            if image is None:
                print(f"Cannot read image current_index:{current_index}")
                continue
            if grayscale_mode:
                image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                cv2.imshow('Image', image)
            else:
                cv2.imshow('Image', image)

            key = cv2.waitKey(1)
            if key == ESC_KEY or key == Q_KEY:
                break
```

```
            if key == ord('g'):
                grayscale_mode = not grayscale_mode
            if key == ord('s') and save_press < self.num_img:
                cv2.imwrite(osp.join(self.save_dir, str(save_press) +'.png'), image)
                save_press += 1

            current_index = (current_index + 1) % len(frames)

        cv2.destroyAllWindows()
```

2 Calibration d'une caméra en OpenCV

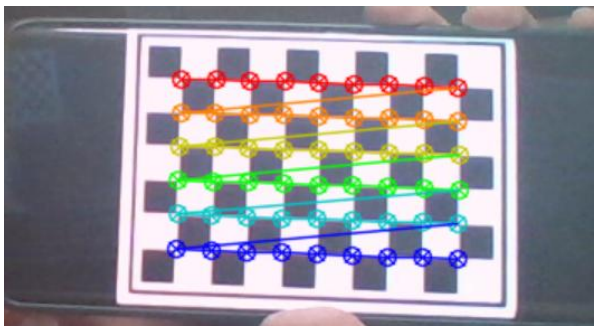
2.1 Détection d'un échiquier

Nous vérifions d'abord si l'image entrée est un graphique en niveaux de gris. Lorsque l'échiquier est effectivement trouvé par findChessboardCorners(), nous affinerons la position des coins en utilisant la méthode cornerSubPix du module imgproc. Afin de vérifier la bonne détection de l'échiquier, nous allons utiliser la méthode drawChessboardCorners du module calib3d (resp. cv2) qui nous permet d'afficher les coins détectés dans une image.

```
def find_corners(self, img):
        if len(img.shape) == 3 and img.shape[2] == 3:
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        else:
            gray = img
        ret, corners = cv2.findChessboardCorners(gray, self.pattern_size, self.flags)
        if ret:
            corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), self.criteria)
            cv2.drawChessboardCorners(img, self.pattern_size, corners2, ret)
        else:
            cv2.putText(img, "Unable to Detect Chessboard", (20, img.shape[0] - 20),
cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 255), 3)

        return img, ret, corners
```

On reconnaît de manière stable tous les treillis du plateau.



2.2 Calibration de la caméra

Les images de la caméra sont d'abord acceptés comme entrée. Un tableau objp de points d'objets tridimensionnels est alors généré en fonction de la taille du motif donné. Les points tridimensionnels dans l'espace du monde réel et les points bidimensionnels dans le plan de l'image sont ensuite stockés. Pour chaque image saisie, appelez la fonction find_corners pour trouver les coordonnées des points angulaires dans l'image et ajoutez ces coordonnées aux tableaux objpoints et imgpoints. Enfin la fonction calibratecamera est appelée pour effectuer le calibrage de

la caméra et retourne le résultat du calibrage de la caméra.

```
    def calibrate_camera(self, img_frames, pattern_size=None):
        h, v = self.pattern_size if pattern_size == None else pattern_size
        objp = np.zeros((v*h,3), np.float32)
        objp[:,:2] = np.mgrid[0:h,0:v].T.reshape(-1,2)

        objpoints = [] # 3d point in real world space
        imgpoints = [] # 2d points in image plane.
        for img in img_frames:
            img, ret_f, corners = self.find_corners(img)
            if ret_f:
                objpoints.append(objp)
                imgpoints.append(corners)
        gray = cv2.cvtColor(img_frames[-1], cv2.COLOR_BGR2GRAY)

        return cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None,
None)
```

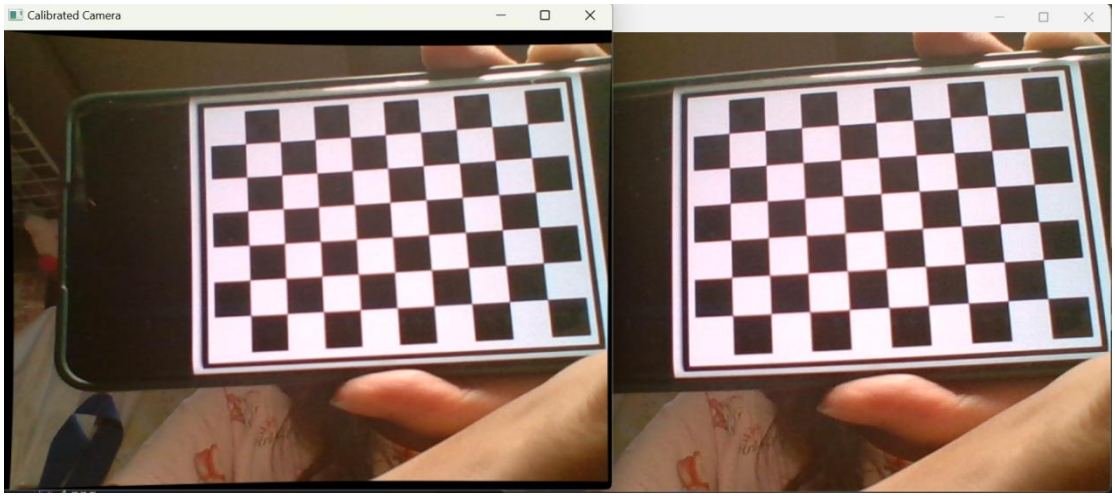
La matrice intrinsèque de la caméra et le coefficient de distorsion que nous obtenons en sortie sont:

```
Image acquisition completed. Starting Camera Calibration!
[[610.96781004   0.         302.66200574]
 [  0.         609.7723435  233.45597136]
 [  0.           0.           1.        ]] [[-3.08111957e-01  1.42312296e+00 -1.69099608e-03  5.36163178e-03
  -2.59116050e+00]]
```

2.3 Redressement de l'image

Nous utilisons la fonction cv2.undistort() pour afficher l'image calibrée.

Après correction, les lignes sont devenues plus droites (à gauche).



2.4 Mise en forme de votre code

Cette code demande des paramètres à l'utilisateur. Demandez à l'utilisateur

d’entrer l’id de la caméra et vérifiez qu’il est correct. L’utilisateur est invité à entrer:

- L’identifiant de la caméra utilisée ;
- le nombre de coin intérieur à l’échiquier dans la largeur ;
- le nombre de coin intérieur à l’échiquier dans la hauteur ;
- le nombre d’images à utiliser pour calculer la matrice intrinsèque et les coefficients

La correction de la caméra se fera automatiquement après que le client ait pris un nombre prédéterminé de photos. Tous les paramètres ci-dessus sont vérifiés: si l’utilisateur a tapé un autre nombre entier, l’utilisateur est invité à taper à nouveau.

```
def user_api():
    while True:
        camera_id = input("L'identifiant de la caméra utilisée:\n")
        if camera_id == '-1':
            exit()
        try:
            camera_id = int(camera_id)
        except ValueError:
            print("Veuillez entrer un nombre entier!")
            continue
        cap = cv2.VideoCapture(camera_id)
        if not cap.isOpened():
            print("ID de caméra incorrect")
            continue
        else:
            print("Successfully open your camera!")
            break
    while True:
        coin_l = input("Le nombre de coin intérieur à l'échiquier dans la largeur:\n")
        try:
            coin_l = int(coin_l)
            break
        except ValueError:
            print("Veuillez entrer un nombre entier!")
            continue
    while True:
        coin_h = input("Le nombre de coin intérieur à l'échiquier dans la hauteur:\n")
        try:
            coin_h = int(coin_h)
            break
        except ValueError:
            print("Veuillez entrer un nombre entier!")
```

```
            continue
    while True:
        num_img = input("Le nombre d'images pour calculer la matrice intrinsèque et les coefficients de distorsion:\n")
        try:
            num_img = int(num_img)
            break
        except ValueError:
            print("Veuillez entrer un nombre entier!")
            continue
    return camera_id, coin_l, coin_h, num_img
```

Interface en cours d'exécution:

```
L'identifiant de la caméra utilisée:
0
Successfully open your camera!
Le nombre de coin intérieur à l'échiquier dans la largeur:
9
Le nombre de coin intérieur à l'échiquier dans la hauteur:
6
Le nombre d'images pour calculer la matrice intrinsèque et les coefficients de distorsion:
20
Dispaly video in loop (Enter a) OR Open camera (ENTER b)?
b
Image acquisition completed. Starting Camera Calibration!
The rest of the image has been calibrated for distortion.
```

3 Calibration d'une caméra en OpenCV à partir d'un ensemble d'images

Les matrices intrinsèques (intrinsic) et les coefficients de distorsion (distCoeffs) du Gopro sont:

```
[[355.37296564   0.          320.79801786]
 [  0.         352.64522555 221.48210655]
 [  0.           0.           1.        ]]
```

et

```
[[-0.52283177  0.35264667  0.00799572  0.0025507  -0.11441567]]
```

Nous obtenons le résultat ci-dessus avec le code suivant:

```
file_dir = "calib_gopro"
img_frames = read_images_from_folder(file_dir)
ret, mtx, dist, rvecs, tvecs = app.calibrate_camera(img_frames)
print(mtx,"\n", dist)
```

Dont:

```
app = DisplayAPP(save_dir,
                 camera_id=camera_id,
                 coin_l=coin_l,
                 coin_h=coin_h,
                 num_img=num_img)
```

Nos méthodes sont toutes encapsulées dans une classe “DisplayAPP”.

Les images avant et après l’étalonnage sont: