Calibration d'une caméra avec OpenCV

Tutoriel complet : https://learnopencv.com/camera-calibration-using-opencv/

Objectif du tutoriel :

L'objectif de ce tutoriel est d'obtenir la matrice d'une caméra. Cela peut notamment servir pour positionner des April Tags (ou tags ArUco) dans l'espace.

On obtient une matrice de ce type :

```python
fx 0 cx
0 fy cy
0  0 1
```

avec :

`fx`, `fy` les largeurs des focales selon les axes x et y en **pixels**

`cx`, `cy` les coordonnées en **pixels** du "point principal" (le point où tous les rayons convergent)

On peut également récupérer la matrice de distorsion

Etape 1 :

Trouver un damier (ou imprimer celui en annexe) et le fixer contre un mur ou un tableau.

Etape 2 :

Prendre des photos du damier dans différentes positions et angles avec la caméra à calibrer.

(utiliser le script photos_damier.py disponible dan le dépot git ou ci-dessous)

Exemple de code python utilisant Opencv pour prendre une série de photos avec une caméra

```
import cv2
import time

cap = cv2.VideoCapture("/dev/video0") # entrer ici le flux vidéo

c=0 # compteur de boucles
d=0 # compteur de photos prises
while d<=8: #régler ici le nombre de photos voulues
   c=c+1
   flag, img = cap.read()
   cv2.waitKey(20)
   cv2.imshow('video',img)
   if c%100 == 0 : #régler ici le nombre de boucles entre chaque photos
       d=d+1
       cv2.imwrite("damier_" + str(d) + ".png",img) # sauvegarde la capture
dans le dossier du programme
       print ("photo " + str(d) + "prise")
```

Etape 3 :

Ouvrir le script camera_calibration.py (disponible dans le dépôt git ou ci-dessous).
Entrer dans la variable images le chemin du dossier contenant les photos du damier.
Exécuter le code et relever la matrice de la caméra et la matrice de distorsion.

```
import numpy as np
import cv2
import glob
# termination criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((9*6,3), np.float32)
objp[:,:2] = np.mgrid[0:9,0:6].T.reshape(-1,2)
# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.
images =
glob.glob("//home/pc-techlab-ia-2/Documents/TOSELLO/projet_video/damiers2/
*.png")
for fname in images:
   img = cv2.imread(fname)
   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
   # Find the chess board corners
   ret, corners = cv2.findChessboardCorners(gray, (9,6), None)
   # If found, add object points, image points (after refining them)
   if ret == True:
       objpoints.append(objp)
       corners2=cv2.cornerSubPix(gray,corners, (11,11), (-1,-1), criteria)
       imgpoints.append(corners)
       # Draw and display the corners
       cv2.drawChessboardCorners(img, (9,6), corners2, ret)
   cv2.imshow('img', img)
   cv2.waitKey(500)
cv2.destroyAllWindows()
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
gray.shape[::-1], None, None)
print(mtx)
print(dist)
```