Master en Intelligence Artificielle et Réalité Virtuelle

Module: Vision par ordinateur

TP1 - Calibration des caméras

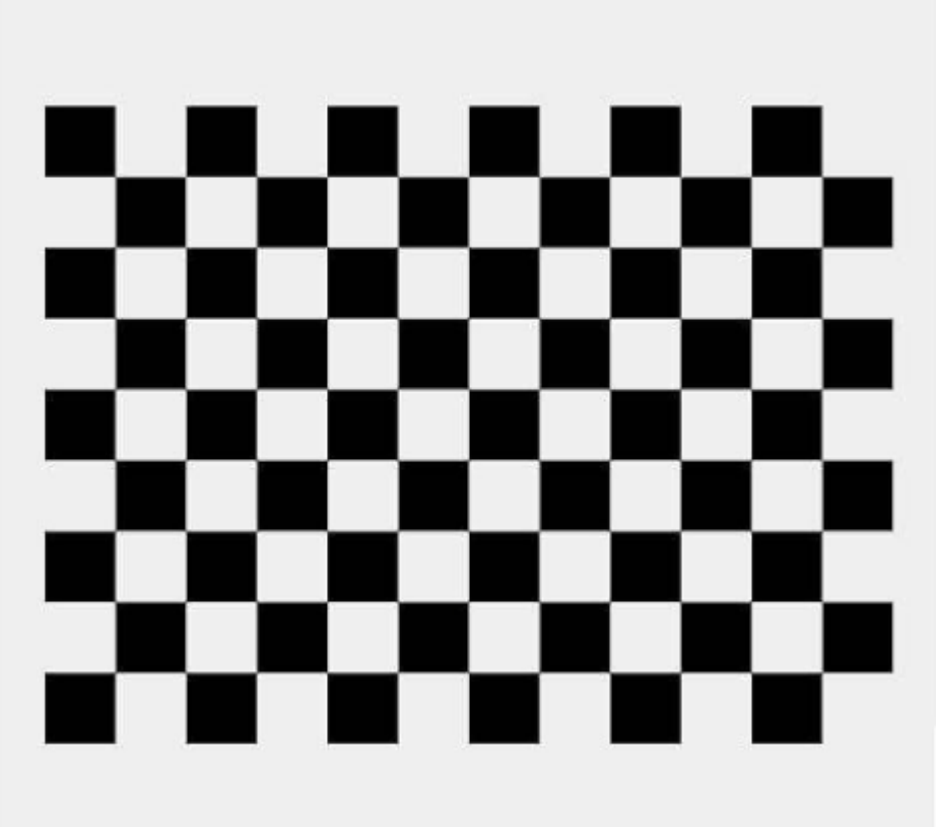
Objectif du TP :

1. Comprendre les principes de base de la **calibration des caméras**.
2. Utiliser des images d'une grille ou d'un damier pour calibrer une caméra.
3. Corriger les erreurs de **distorsion radiale et tangente**.
4. Visualiser l'effet de la calibration sur les images.

Matériel requis :

- Ordinateur avec Python installé.
- Bibliothèques Python : **OpenCV** et **NumPy**.
- Jeu d'images d'une **grille de calibration** ou d'un **damier**
- Caméra ou vidéo enregistrée d'un damier (optionnel si des images sont déjà fournies).

Exemple d’un damier :



Énoncé de TP :

1. Introduction à la calibration des caméras :

Avant de plonger dans le codage, répondez aux questions théoriques suivantes :

- Quels sont les **paramètres intrinsèques** d'une caméra ?
- Quels sont les **paramètres extrinsèques** d'une caméra ?
- Quelles sont les principales distorsions qui affectent une caméra (radiale, tangentielle) ?

2. Chargement et préparation des images :

Téléchargez un ensemble d'images d'un damier ou d'une grille de calibration prises sous différents angles avec une caméra fixe. Vous pouvez également capturer ces images vous-même si un matériel de capture est disponible.

3. Détection des coins du damier :

Utilisez la bibliothèque **OpenCV** pour détecter les coins dans les images de calibration.

Exemple : code python

```python
## """ Cours Vision par Ordinateur """
"S3 : Master en Intelligence Artificielle et Réalité Virtuelle"
##
import numpy as np
import cv2
import glob

# Taille du damier (9x6 par exemple)
board_size = (9, 6)

# Préparer des points 3D du monde réel
objp = np.zeros((board_size[0]*board_size[1], 3), np.float32)
objp[:,:2] = np.mgrid[0:board_size[0], 0:board_size[1]].T.reshape(-1, 2)

# Liste pour stocker les points 3D et 2D
objpoints = [] # Points 3D du monde réel
imgpoints = [] # Points 2D dans l'image

# Charger toutes les images du damier
images = glob.glob('C:/NMO/cours_computer_vision/TP1/calibration/images/*.jpg')

for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Détecter les coins du damier
    ret, corners = cv2.findChessboardCorners(gray, board_size, None)

    # Si les coins sont trouvés, ajouter les points 3D et 2D
    if ret == True:
        objpoints.append(objp)
        imgpoints.append(corners)

        # Dessiner les coins
        cv2.drawChessboardCorners(img, board_size, corners, ret)
        cv2.imshow('img', img)
        cv2.waitKey(500)

cv2.destroyAllWindows()
```

4. Calibration de la caméra :

Utilisez les points détectés pour calibrer la caméra. Cette étape vous donnera les **paramètres intrinsèques** (matrice de la caméra) et les **coefficients de distorsion**.

```python

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None, None)

print("Matrice de la caméra :\n", mtx)
print("Coefficients de distorsion :\n", dist)

```

5. Correction de la distorsion :

Une fois la calibration effectuée, utilisez les paramètres trouvés pour corriger les images et supprimer les distorsions.

```
# Charger une image distordue
img = cv2.imread('path/to/your/image.jpg')
h, w = img.shape[:2]
newcameramtx, roi = cv2.getOptimalNewCameraMatrix(mtx, dist, (w,h), 1, (w,h))

# Corriger la distorsion
dst = cv2.undistort(img, mtx, dist, None, newcameramtx)

# Rogner l'image
x, y, w, h = roi
dst = dst[y:y+h, x:x+w]
cv2.imwrite('calibrated_result.jpg', dst)

```

6. Analyse des résultats :

- Comparez les images avant et après la correction de la distorsion.
- Mesurez les différences observées visuellement et discutez des erreurs restantes.

7. Évaluation de la calibration :

Utilisez la fonction suivante pour évaluer l'erreur de reprojection, ce qui permet de mesurer la précision de votre calibration.

```
mean_error = 0
for i in range(len(objpoints)):
    imgpoints2, _ = cv2.projectPoints(objpoints[i], rvecs[i], tvecs[i], mtx, dist)
    error = cv2.norm(imgpoints[i], imgpoints2, cv2.NORM_L2)/len(imgpoints2)
    mean_error += error

print(f"Erreur totale de reprojection : {mean_error/len(objpoints)}")

```

Compte rendu attendus :

- Code source utilisé pour calibrer la caméra.
- **Matrice de la caméra** : matrice intrinsèque obtenue après la calibration.
- **Coefficients de distorsion** : valeurs qui définissent les distorsions radiales et tangentielles de la caméra.
- **Image corrigée** : image testée après correction de la distorsion.
- Rapport sur l'erreur de reprojection et les résultats obtenus.