Perception 3D

I. Mise en place de l'environnement

Les TPs nécessitent l'utilisation d'une webcam et d'un environnement de développement python3.11 avec les paquets OpenCV-4, Open3d et matplotlib.

Vérification des dépendances:

- Tester le bon fonctionnement d'opencv, open3d et matplotlib en exécutant les scripts : `opencv_test.py`, `open3d_test.py` et `matplotlib_test.py`

II. Vision Monoculaire

1. Calibration

Dans cette partie, nous allons développer un programme dédié à la calibration d'une caméra à l'aide de la bibliothèque OpenCV.

Nous allons commencer par acquérir des images de la mire :

1. Implémenter la fonction de détection adéquate dans la méthode `CalibrationBase.detect()` du fichier `Calibration.py`. Quel type de mire avez-vous à disposition ? Comment fonctionne la détection de ce type de mire ?
- Instancier un objet `MonoCalibration` dans le fichier `MonoMain.py`. Utiliser la fonction `acquire()` afin d'acquérir des images de la mire. Appuyer sur la touche *'space'* ou *'enter'* afin de sauvegarder une image. Lorsque suffisamment d'images ont été acquises, fermer l'application en appuyant sur *'escape'*. Vous pouvez retrouver les images enregistrées dans le dossier `results`.

Nous allons maintenant réaliser la calibration de la caméra à partir des images acquises précédemment :

2. À quoi correspondent les variables `objectPoints` et `imgPoints` dans la fonction `CalibrationBase.detectInImages()` ?
- Implémenter la fonction `calibrateCameraExtended()` à l'endroit indiqué dans la méthode `MonoCalibration.calibrate()`.
3. Comment sont obtenues les valeurs intrinsèques, extrinsèques et les coefficients de distorsions ? À quoi sert le paramètre `squareSize_m` ?

Nous allons ensuite analyser la calibration et, si besoin, tenter d'améliorer les résultats obtenus précédemment :

4. Utiliser les fonctions `visualizeBoards()` et `plotRMS()`, et analyser les valeurs intrinsèques, les coefficients de distorsion ainsi que le RMS. Que représentent les figures obtenues ? Que peut-on déduire sur la qualité de la calibration ? Quels sont les facteurs importants pour réaliser un étalonnage précis ?

2. Rectification

Dans cette partie, nous allons procéder à la rectification des images d'une caméra à l'aide de la bibliothèque OpenCV :

6. Ouvrir le fichier `Rectification.py` et implémenter les fonctions manquantes dans `MonoRectification.computeCorrectionMaps()`. Qu'est-ce qu'une 'correction map' ?
- Implémenter la fonction `remap()` dans la méthode `MonoRectification.rectify()`.
- Instancier un objet `Rectification` dans le fichier `MonoMain.py`.
- Utiliser la fonction `display()` afin de visualiser en direct les images rectifiées de votre caméra.

7. Faire un screenshot. Quel est le type de distorsion ? Comment s'observe-t-il dans l'image ?

III. Stéréovision

1. Calibration

Dans cette partie, nous allons calibrer un banc stéréo à partir d'images préenregistrées. Les cellules de la mire utilisée pour cette calibration font ***108 mm***.

- Implémenter `stereoCalibrateExtended()` dans la méthode `StereoCalibration.Calibrate()` du fichier `Calibration.py`.
- Instancier un objet `StereoCalibration` dans le fichier `StereoMain.py`.

8. Procéder à la calibration, utiliser les fonctions `visualizeBoards()` et `plotRMS()`, et analyser les valeurs intrinsèques, les coefficients de distorsion ainsi que le RMS. Que peut-on déduire sur la qualité de la calibration ? Les valeurs semblent elles cohérentes ?

2. Rectification

Une fois l'étalonnage effectué nous allons procéder à la stéréo-rectification :

- Une fonction d'OpenCV permet de réaliser cette opération, l'implémenter dans la fonction `StereoRectification.computeCorrectionMaps()` du fichier `Rectification.py`.
- Instancier un objet `StereoRectification` dans le fichier `StereoMain.py`.
- Utiliser la fonction `display()` pour visualiser des paires d'images rectifiées.

9. Afficher les lignes épipolaires, que peut-on en conclure quant à la qualité de la calibration ?

3. Reconstruction 3D

La dernière étape consiste à reconstruire l'environnement en 3D :

10. Compléter le code de la fonction `StereoRectification.displayDisparity()` afin de calculer une carte de disparité. À quoi correspondent les valeurs de cette carte de disparité ?
11. Visualiser l'image de disparité. Faire varier les différents paramètres. Faire un screenshot.
12. Comment peut-on obtenir une carte de profondeur à partir d'une image de disparité ? Implémenter une fonction permettant de calculer cette carte de profondeur et de la visualiser. Faire un screenshot.
13. Comment reprojeter une carte de profondeur 2D en un nuage de points 3D ? Implémenter une fonction permettant d'obtenir ce nuage de points et de le visualiser (vous pouvez utiliser open3d). Faire un screenshot.