Perception 3D

I. Mise en place de l'environnement

Ce TP est à réaliser sous environnement linux et nécessite l'utilisation d'une webcam et d'un environnement de développement python3 avec les paquets OpenCV et matplotlib.

Déploiement de l'environnement de développement :

- Créer un dossier `tp-per-env`
- Extraire l'environnement de développement :
  `tar -xzf tp-per-env.tar.gz -C tp-per-env`
- Activer l'environnement :
  `source tp-per-env/bin/activate`
  `conda-unpack`
- Tester le bon déploiement de l'environnement en exécutant les scripts : `opencv_test.py` et `matplotlib_test.py`

II. Vision Monoculaire

1. Calibration

Dans cette partie, nous allons développer un programme dédié à la calibration (ou aussi étalonnage) d'une caméra monoculaire à l'aide de la bibliothèque Open CV.

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 ?
2. Utiliser la fonction `Calibration.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 :

3. À quoi correspondent les variables `objectPoints` et `imgPoints` dans la fonction `CalibrationBase.detectInImages()` ?
4. Implémenter la fonction `calibrateCameraExtended()` à l'endroit indiqué dans la méthode `MonoCalibration.calibrate()`.
5. Comment sont obtenues les valeurs intrinsèques, extrinsèques et les distorsions ?
6. Quelles sont les valeurs intrinsèques, les coefficients de distorsion ainsi que le RMS retournés ? À quoi correspond le RMS ? Que peut-on en déduire sur la qualité de la calibration ?

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

7. Utiliser la fonction `MonoCalibration.visualizeBoards()`. Que représente la figure obtenue ? Que peut-on en déduire ?
8. Utiliser la fonction `MonoCalibration.plotRMS()`. Que représente la figure obtenue ? Que peut-on en déduire ?

9. Proposer et appliquer une/des méthode(s) afin d'améliorer les résultats de la calibration en se basant sur les observations des points précédents. Quels sont les facteurs importants pour réaliser un étalonnage précis ?
10. Quelles sont les valeurs intrinsèques, les coefficients de distorsion ainsi que le RMS retournés ?

2. Rectification

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

11. Ouvrir le fichier `Rectification.py` et implémenter les fonctions manquantes dans `MonoRectification.computeCorrectionMaps()`. Qu'est-ce qu'une 'correction map' ?
12. Implémenter la fonction `remap()` dans la méthode `MonoRectification.rectify()`.
13. Utiliser la fonction `MonoRectification.display()` afin de visualiser en direct les images rectifiées de votre caméra.
14. Quel est le type de distorsion ? Comment s'observe-t-il dans l'image ?
15. A l'aide de l'ensemble des résultats obtenus dans cette partie, que peut-on en conclure sur la caméra et son optique ?

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 *108mm*.

16. Implémenter `stereoCalibrateExtended()` dans la méthode `StereoCalibration.Calibrate()` du fichier `Calibration.py`. Comment fonctionne cette fonction ?
17. Procéder à la calibration. Quelles sont les valeurs des paramètres intrinsèques, extrinsèques, des coefficients de distorsion et du RMS ? Les valeurs semblent elles cohérentes ?
18. Afficher le graphe du RMS. Que peut-on en déduire ? Prendre les actions nécessaires afin d'améliorer les résultats de la calibration. Pourquoi certaines images posent-elle problème ?

2. Rectification

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

19. Quel est l'intérêt de la stéréo-rectification ? Une fonction d'OpenCV permet de réaliser cette opération, l'implémenter dans la fonction `StereoRectification.computeCorrectionMaps()` du fichier `Rectification.py`.
20. Utiliser la fonction `StereoRectification.display()` pour visualiser des paires d'images rectifiées. Qu'observe-t-on ?
21. 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 :

22. 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 ?
23. Visualiser l'image de disparité. Faire varier les différents paramètres.
24. Comment peut-on obtenir une carte de profondeur à partir d'une carte de disparité ?
25. Comment reprojeter une carte de profondeur 2D en un nuage de points 3D ?