

Perception 3D

I. 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. Quels sont les paramètres de cette fonction ? Comment sont obtenues les valeurs intrinsèques, extrinsèques et les distorsions ?
6. Quelles sont les valeurs intrinsèques, les coefficients de distortion 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 méthodes 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*. À quoi servent ces deux fonctions ? 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. À quoi servent les différents paramètres ?
14. Quel est le type de distorsion ? Comment s'observe-t-il dans l'image ?
15. À l'aide de l'ensemble des résultats obtenus dans cette partie, que peut-on en conclure sur la caméra et son optique ?

II. 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*. Quels sont ses paramètres ? Comment fonctionne-t-elle ?
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-elles problème ?

2. Rectification

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

18. Quel est l'intérêt de la stéréo-rectification ? Quelle fonction d'OpenCV permet de réaliser cette opération ? Quels sont ses arguments ? L'implémenter dans la fonction *StereoRectification.computeCorrectionMaps* du fichier *Rectification.py*.
19. Utiliser la fonction *StereoRectification.display* pour visualiser des paires d'images rectifiées. Qu'observe-t-on ? Proposer une explication au phénomène observé. Quelle solution peut-on mettre en œuvre ?
20. 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 :

21. Compléter le code de la fonction *StereoRectification.displayDisparity* afin de calculer une carte de disparité. Quelle est la méthode utilisée ? À quoi correspondent les valeurs de cette carte de disparité ?
22. Visualiser l'image de disparité. Faire varier les différents paramètres. À quoi servent-ils ?
23. Comment peut-on obtenir une carte de profondeur à partir d'une image de disparité ?
24. Comment reprojeter une carte de profondeur 2D en un nuage de points 3D ?