ESIREM
ÉCOLE SUPÉRIEURE D'INGÉNIEURS
NUMÉRIQUE ET MATÉRIAUX

Modélisation 3D

Compte rendu du TP Reconstruction 3D

TEBAI Osama

Sommaire

Introduction

Le but de ce TP est d'apprendre à reconstituer une image 3D, notamment l'acquisition des résultats, du calibrage et de la carte de profondeur. Nous allons pouvoir retrouver le calibrage de l'appareil photo utilisé pour prendre les photos à l'aide des matrices calculés par openCV.

A cause d'un manque de temps, que la première partie a été effectué, et les images utilisés sont les images fournis par le professeur.

Calibrage de la caméra

Estimation de la matrice de la caméra

Après avoir téléchargé le fichier modelisation.py, on s'intéresse à la fonction CameraCalibration().

Nous avons modifié les chemins pour les associer aux bons répertoires, et ensuite nous avons lancé le code pour un premier test. Lors du premier lancement, nous voyons les images défiler une après les autres, et s'arrêter lors de la dernière. On remarque que dans ces images, nous trouvons des lignes qui relient des points de l'échiquier. En regardant le code, nous avons compris que dans la fonction CameraCalibration(), on cherche les points d'intersection entre les cases blanches et les cases noires. Les lignes qu'on voit sont des lignes qui relient tous les points retrouvés.

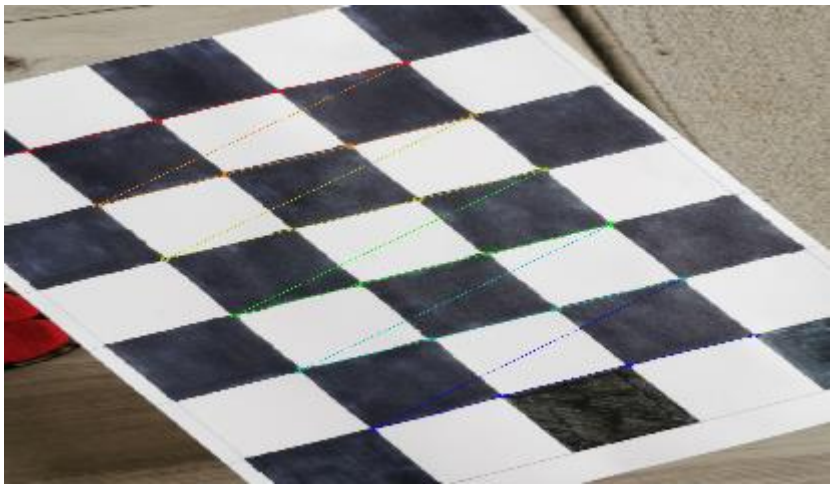

Figure 1 : Lignes qui relient les points d'intersection

On remarque aussi que dans la console, nous avons des booléens qui sont affichés à la suite. En cherchant dans le code, on remarque que lorsque CameraCalibration trouve les points d'intersection, il affiche l'image et il renvoie « True », sinon, s'il ne trouve pas de points, il revoie « False » sans afficher d'images.

Analyse des résultats

On remarque que les photos 2D possèdent des déformations au niveau des bords. Nous avons déduit que la cause de ce phénomène est la distorsion en coussinet. En effet on observe l'effet fish-eye, qui grossit la zone centrale de l'image.

La console, après avoir envoyé les valeurs booléens, nous affiche des matrices. Ces matrices sont obtenues après la calibration, et elles représentent les paramètres intrinsèques et extrinsèques de la caméra.

```
camraMatrix
 [[1.41648092e+03 0.00000000e+00 6.67598684e+02]
 [0.00000000e+00 1.41360922e+03 9.16650420e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
dist
 [[-0.00996315  0.66100361  0.00470359 -0.00439002 -1.82246334]]
newcameramtx
 [[1.00423413e+03 0.00000000e+00 5.52873962e+02]
 [0.00000000e+00 7.90379700e+02 1.13082257e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

Figure 2 : Matrices de calibration

La première matrice, « camraMatrix », représente les paramètres intrinsèques et extrinsèques de la caméra calculés par openCV. La deuxième matrice, « newcameramtx », est une matrice qui est calculé grâce à la fonction de openCV « getOptimalNewCameraMatrix() ».

Une fois que nous avons obtenu ces deux matrices, le script calcul l'image rectifiée, que dans notre cas est la suivante :

Figure 3 : image rectifiée par le script

Comme on peut voir de la figure 3, l'effet fish-eye est disparu par rapport aux images de openCV.