

**Analyse d’Image :**

**Reconstruction 3D**

Cassandra Breton, Julien Fleckinger

et Miguel Reuter

# Projet : calibrage d’un système stéréoscopique

## (But : Trouver les coefficients de distorsion et corriger l’image)

Pour commencer, nous avons créé un panel de 25 images d’un damier de 8x5 cases, dont chaque case fait 35.25x35.25mm. L’appareil utilisé pour la prise des photos crée un effet de « fisheye » sur l’image, ainsi les corrections se voient beaucoup mieux.

Nous obtenons 7x4 points d’intérêts (correspondants aux intersections) par image.

Pour chaque prise, nous avons fait varier la position du damier par rapport à la caméra.

L’intégralité du code se trouve dans « Reconstruction3D\src\main.cpp ».

Image avant calibrage :



Voici l’image obtenu après correction :



# Initialisation des paramètres

Cette partie va détailler les paramètres intrinsèques, demandé à l’utilisateur.

Ces paramètres sont nécessaires au paramétrage de la caméra. Ils se trouvent dans le fichier « Reconstruction3D\conf\in\_VID5.xml » et « FISHEYE.xml ».

**BoardSize\_Width** -> Le nombre de points d’intérêt que contient le damier présent sur les images, en largeur. **Valeur =** 7

**BoardSize\_Height** -> Le nombre de points d’intérêt que contient le damier présent sur les images, en hauteur. **Valeur =** 4

**Square\_Size** -> La taille des cases du damier en millimètres. **Valeur =** 35.25

**Input** -> Contient le chemin vers le fichier de configuration où se trouve la liste des images à utiliser. **Valeur =** «Reconstruction3D\ conf\FISHEYE.xml »

**Input\_Delay** -> Délai entre chaque changement d’image, lors de la calibration, en millisecondes. **Valeur =** 100

**Calibrate\_FixAspectRatio** -> Demande à ce que l’aspect ratio soit le même pour fx et fy que dans l’input cameraMatrix. **Valeur =** 1

**Calibrate\_AssumeZeroTangentialDistortion** -> Oblige les coefficients de distorsion tangentielle à être égale à 0. **Valeur =** 0

**Calibrate\_FixPrincipalPointAtTheCenter** -> Permet de corriger le point d’intérêt principale. **Valeur =** 1

**Write\_outputFileName** -> Fichier où sera enregistré les paramètres de calibration de la caméra. **Valeur =** « out\_camera\_data.xml »

# Méthodes de calculs des coefficients et correction

Cette partie traite des méthodes implémentés dans le code. Celui-ci se trouve dans « Reconstruction3D\src\main.cpp ».

## Autres méthodes :

**Settings::read(const FileNode & f)** -> Permet de lire les données enregistrées dans le fichier f de paramétrage et utilise interprate() pour initialisé le programme avec.

**Settings::interprate()** -> Fait des tests de validité des paramètres donnés en entrée et initialise le programme avec.

**Settings::Mat nextImage()** -> Donne la matrice contenant l’image suivante.

**bool Settings::readStringList( const string& f, vector<string>& l )** -> Permet de lire le fichier contenant la liste des images f et met les chemins des images dans la liste l.

**bool Settings::isListOfImages( const string& f)** -> Vérifie que le fichier fichier f (contenant la liste des chemins d’images) est accepté par le programme.

**static double computeReprojectionErrors( const vector<vector<Point3f> >& points, const vector<vector<Point2f> >& p\_image, const vector<Mat>& rvecs, const vector<Mat>& tvecs, const Mat& m\_camera, const Mat& distCoeffs, <float>& perViewErrors)** ->

**read (const FileNode& f, Settings& s, const Settings& default = Settings())** -> Initialise la structure Settings s , soit avec des données récupérées dans le FileNode f, soit avec une structure par défaut.

**calcBoardCornerPositions(Size s\_boardSize, float s\_square, vector<Point3f>& corners)** -> Fonction permettant de trouver les points d’intérêts sur l’image.

**saveCameraParams(Settings& s, Size& s\_image, Mat& m\_camera, Mat& distCoeffs, const vector<Mat>& rvecs, const vector<Mat>& tvecs, const vector<float>& reprojErrs, const vector<vector<Point2f> >& imagePoints, double totalAvgErr)** -> Permet d’enregistrer les paramètres calculés de la caméra dans un fichier afin de pouvoir les réutiliser.

**bool runCalibrationAndSave(Settings& s, Size imageSize, Mat& cameraMatrix, Mat& distCoeffs,vector<vector<Point2f> > imagePoints )** -> Lance runCalibration() et vérifie si les résultats sont bons pour savoir s’il est nécessaire d’enregistrer les paramètres calculés dans un fichier.