Projet imagerie 3D : Reconstruction d'une scène à partir d'images 2D

telecom saint-étienne
école d'ingénieurs nouvelles technologies

Duteyrat Antoine, Sève Léo

---

5 juin 2025

Table des matières

1 Objectif

L'objectif de ce projet est de reconstruire une scène 3D à partir d'images 2D prises d'une caméra arbitraire. Pour cela, plusieurs étapes sont nécessaires :

- Calibrer la caméra (calibraton intrinsèque) par la méthode de Zhang (mire plane).

- Prendre plusieurs images d'une scène 3D avec la caméra.

- Triangulation des points 3D à partir des images 2D.

2 Calibration de la caméra

2.1 Obtention de la matrice intrinsèque K

La calibration de la caméra est une étape cruciale pour obtenir des images 3D précises. Elle permet de déterminer la matrice intrinsèque K, qui est utilisée pour projeter les points 3D du monde réel (repère caméra) sur l'image 2D capturée par la caméra (figure 1).

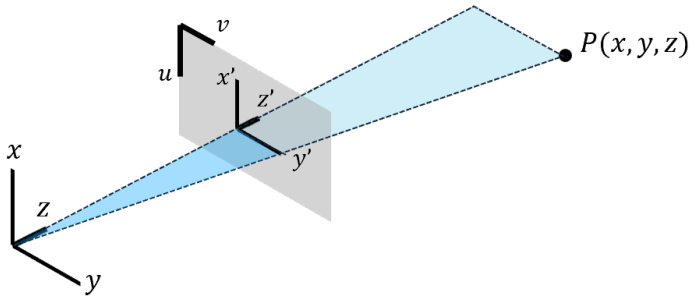

FIGURE 1 – Illustration de la projection des coordonnées 3D du repère caméra sur l'image

Les distances entre les plans sont les suivantes :

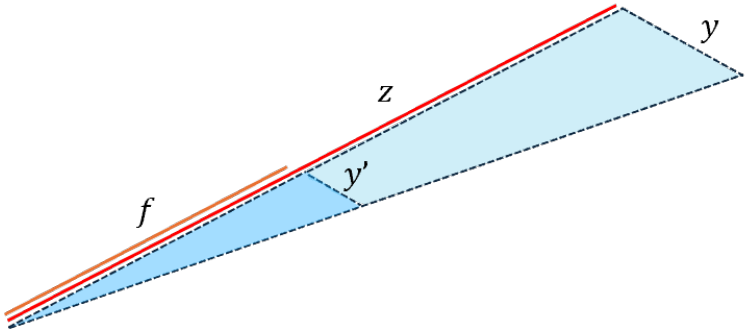

FIGURE 2 – Illustration des distances entre les plans de la caméra

De cette illustration, on tire les égalités suivantes :

$$\frac{y'}{y} = \frac{f}{z} \quad \text{et} \quad \frac{x'}{x} = \frac{f}{z} \tag{1}$$

où :

— x' et y' sont les coordonnées de l'image (en mm),
— x, y et z sont les coordonnées du repère caméra (en mm),
— f est la distance focale de la caméra (en mm).

D'où :

$$y' = \frac{f \times y}{z} \quad \text{et} \quad x' = \frac{f \times x}{z} \tag{2}$$

L'objectif est maintenant de passer des coordonnées en mm du repère $x'; y'$ au repère $u; v$ de l'image (respectivement vertical et horizontal). Pour cela, on introduit k_u et k_v les facteurs d'échelle (en pixels/mm), et on ajoute le décalage u_0 et v_0 (en pixels) pour obtenir les coordonnées en pixels dans l'image.

$$u = -k_u \times x' + u_0 = \frac{-k_u \times f \times x}{z} + u_0 \quad \text{et} \quad v = k_v \times y' + v_0 = \frac{k_v \times f \times y}{z} + v_0 \tag{3}$$

Et en prenant $z = s$:

$$u = \frac{-k_u \times f \times x}{s} + u_0 \quad \text{et} \quad v = \frac{k_v \times f \times y}{s} + v_0 \tag{4}$$

En multipliant par s et en réorganisant, on obtient :

$$\begin{pmatrix} s \times u \\ s \times v \\ s \end{pmatrix} = \begin{pmatrix} -f \times k_u & 0 & u_0 \\ 0 & f \times k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{5}$$

En notant :

— - $f \times k_u = \alpha_u$: distance focale en pixels (en pixels),
— $f \times k_v = \alpha_v$: distance focale en pixels (en pixels),
— $s = z$: coordonnée de la caméra (en mm)

On obtient alors la matrice intrinsèque K de la caméra :

$$\begin{pmatrix} sv \\ su \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{6}$$

Avec la matrice K au centre de l'opération (nous ne considérons pas le cisaillement γ, qui représente l'orthogonalité des lignes et colonnes du capteur de la caméra). Cette matrice est utilisée pour projeter les points 3D du repère caméra sur l'image 2D capturée par la caméra.

2.2 Lien matrices K et A, utilisation et projection

L'objectif de la calibration est d'estimer les valeurs de la matrice intrinsèque K de la caméra de la forme suivante [2][3] (équation 7) :

$$\begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7}$$

où :

- α_u et α_v représentent la distance focale f en pixels dans les directions verticale et horizontale respectivement,
- u_0 et v_0 sont les coordonnées du centre optique en pixels,
- La dernière ligne est utilisée pour homogénéiser les coordonnées.

Cette matrice est ensuite utilisée dans la conversion des coordonnées 3D du monde réel en coordonnées 2D de l'image, en modèle sténopé sans défaut d'orthogonalité (équation 8).

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{8}$$

avec la matrice d'extrinsèque A de la caméra de la forme suivante (équation 9) :

$$A = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \tag{9}$$

Et on note H la matrice d'homographie de la forme suivante (équation 10) :

$$H = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \tag{10}$$

Pour estimer les paramètres de la matrice d'homographie, nous utiliserons la méthode de Zhang. La scène qui nous permettra de calibrer la caméra comportera une mire damier (figure 3).

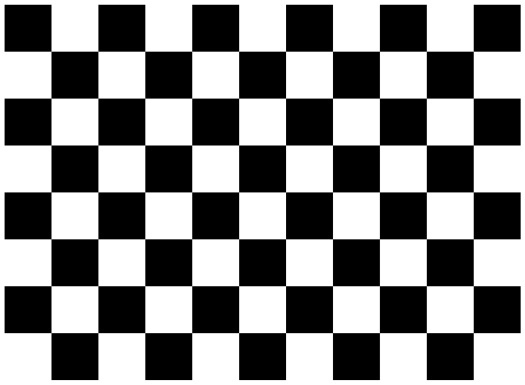

FIGURE 3 – Exemple d'une mire damier

2.3 Application de la méthode Zhang pour estimer K

La méthode de Zhang a été appliquée dans le fichier `src/calib_zhang.py` à partir des informations trouvées dans le papier de Z. Zhang [1].

2.3.1 Calcul de la matrice d'homographie H

Le calcul de la matrice d'homographie H est effectué à partir d'images de la mire damier, prises sous différents angles avec la caméra. La première étape consiste donc à utiliser la fonction `findChessboardCorners` de OpenCV pour détecter les coins de la mire damier dans les images. On récupère ainsi les coordonnées des coins internes de la mire dans l'image (en pixels) et on les stocke dans un tableau. La correspondance entre ces points et des points 3D connus dans le repère monde permet d'utiliser la fonction `findHomography` de OpenCV pour calculer la matrice d'homographie H. Pour rappel, matrice d'homographie H est de la forme suivante (équation 11) :

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \end{pmatrix} \tag{11}$$

Dans la méthode de Zhang, on simplifie cette matrice d'homographie H en taille 3×3 (équation 12) en négligeant la rotation sur l'axe z dans un plan deux dimensions.

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{12}$$

2.3.2 Calcul de la matrice intrinsèque K

Pour estimer les paramètres intrinsèques de la caméra, on s'appuie sur l'algorithme de Zhang : on commence par collecter plusieurs matrices d'homographie $\{H_i\}$ (une par image de la mire) puis on résout un système linéaire de la forme $V\,b = 0$, où b contient les coefficients de la matrice symétrique $B = K^{-T}K^{-1}$. Les étapes sont les suivantes :

1. **Construction de la matrice V**

   Pour chaque homographie H, on calcule deux vecteurs :

   $$v_{01}(H) \quad \text{et} \quad v_{00}(H) - v_{11}(H),$$

   où, pour $i, j \in \{0, 1, 2\}$,

   $$v_{ij}(H) = \Big[h_{1i}h_{1j},\ h_{1i}h_{2j} + h_{2i}h_{1j},\ h_{2i}h_{2j},\ h_{3i}h_{1j} + h_{1i}h_{3j},\ h_{3i}h_{2j} + h_{2i}h_{3j},\ h_{3i}h_{3j}\Big]^T.$$

   On concatène ces lignes dans une grande matrice V.

2. **Décomposition SVD et vecteur solution**

   On récupère la conjuguée-transposée de la matrice V et on effectue une décomposition en valeurs singulières (SVD) grâce à la fonction `np.linalg.svd` de NumPy. On obtient ainsi b avec $b = V^T u$, où u est le dernier vecteur de la SVD de V (correspondant à la plus petite valeur singulière).

3. **Reconstruction de la matrice B**

   On recompose

   $$B = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{pmatrix},$$

   avec $B_{11} = b_0$, $B_{12} = b_1, \ldots, B_{33} = b_8$.

4. **Calcul des paramètres intrinsèques**

   En posant $v_0 = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2}$ et $\lambda = B_{33} - \Big(B_{13}^2 + v_0\,(B_{12}B_{13} - B_{11}B_{23})\Big)/B_{11}$, on obtient :

   $$\alpha_u = \sqrt{\frac{\lambda}{B_{11}}}, \quad \alpha_v = \sqrt{\frac{\lambda\, B_{11}}{B_{11}B_{22} - B_{12}^2}}, \quad \gamma = -\frac{B_{12}\,\alpha_u^2\,\alpha_v}{\lambda}, \quad u_0 = \frac{\gamma\, v_0}{\alpha_v} - \frac{B_{13}\,\alpha_u^2}{\lambda}.$$

5. **Formation de la matrice intrinsèque K**

   Enfin,

$$K = \begin{pmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Ce processus garantit une estimation robuste de la calibration interne de la caméra à partir de jeux de points plans projetés.

2.3.3 Optimisation par minimisation d'une fonction de perte

Cette étape est optionnelle et assez peu détaillée dans le papier de Z. Zhang. Nous avons donc décidé de ne pas l'implémenter dans notre code. Ce choix nous contraint en revanche à utiliser la fonction `calibrateCamera` de OpenCV pour obtenir la matrice intrinsèque K optimisée et les coefficients de distorsion de la caméra (dans le fichier `calib_opencv.py`).

3 Utilisation des intrinsèques caméra K pour la reconstruction 3D

3.1 Matériel et données utilisées

Nous avons utilisé une caméra d'iPhone 13 pour capturer les images de la mire et de l'objet 3D (figure 4 et figure 5).

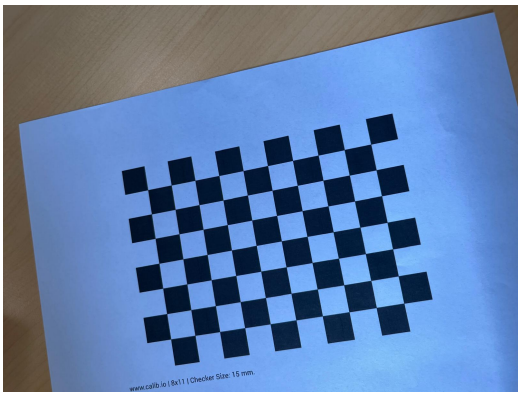

FIGURE 4 – Photo d'une mire prise avec l'iPhone 13

FIGURE 5 – Photo d'un objet 3D pris avec l'iPhone 13

Pour limiter les détections de points d'intérêt sur les motifs derrière l'objet 3D, nous avons utilisé un fond uni (noir) en post-traitement des images d'objet (figure 6).

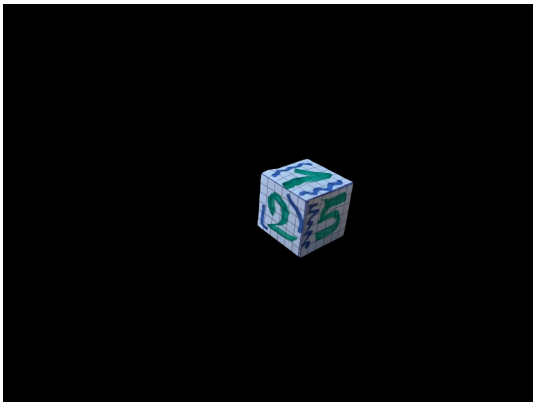

FIGURE 6 – Photo du même objet 3D sans fond

3.2 Reconnaissance des points d'intérêt

Pour la reconnaissance des points d'intérêt dans les images, nous avons utilisé l'algorithme SIFT (Scale-Invariant Feature Transform) de OpenCV. Cet algorithme est robuste aux variations d'échelle et de rotation, et fournit en sortie une liste de points d'intérêt détectés dans l'image (figure 7), ainsi que leurs descripteurs associés (caractéristiques).

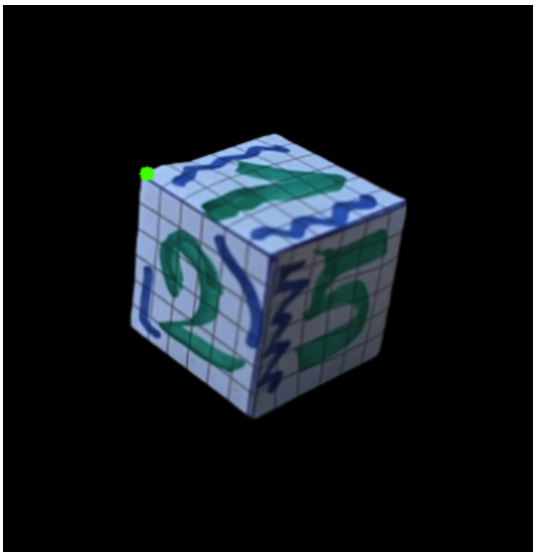

FIGURE 7 – Photo du même objet 3D sans fond avec un exemple de point d'intérêt détecté (cercle vert)

3.3 Correspondance des points d'intérêt dans les paires d'images

Ces points d'intérêt détectés dans les différentes images doivent être mis en correspondance pour permettre la reconstruction 3D. Nous avons ici fait attention de nommer des images prises successivement correctement, pour assurer la présence de paires de points d'intérêt entre les images.
Nous avons utilisé la classe `BFMatcher` de OpenCV pour effectuer cette correspondance. Cette classe permet ensuite d'utiliser la méthode `knnMatch` pour trouver les correspondances entre les points d'intérêt de deux images.
Nous avons fixé le nombre de correspondances à 2 pour chaque point d'intérêt, ce qui permet de trouver les deux points les plus proches dans l'espace des caractéristiques. Pour garder la meilleure correspondance, nous avons utilisé le ratio de Lowe (0.75)

pour filtrer les correspondances. Ce ratio permet de ne garder que les correspondances dont la distance est inférieure à 75% de la distance du deuxième point le plus proche.

3.4 Matrice essentielle et tri des correspondances

L'étape suivante consiste à calculer la matrice essentielle E à partir des correspondances de points d'intérêt. Cette matrice est utilisée ensuite pour estimer la rotation et la translation entre les deux images. Nous avons utilisé la fonction `findEssentialMat` de OpenCV pour calculer la matrice essentielle E à partir des correspondances de points d'intérêt. Le calcul de la fonction donne aussi en sortie un masque de correspondances, qui permet de filtrer les correspondances qui ne sont pas valides, en choisissant la méthode de tri (ici Ransac).
Chaque point d'intérêt d'une image N, à ce moment de l'algorithme, est associé à un point d'intérêt de l'image $N + 1$.

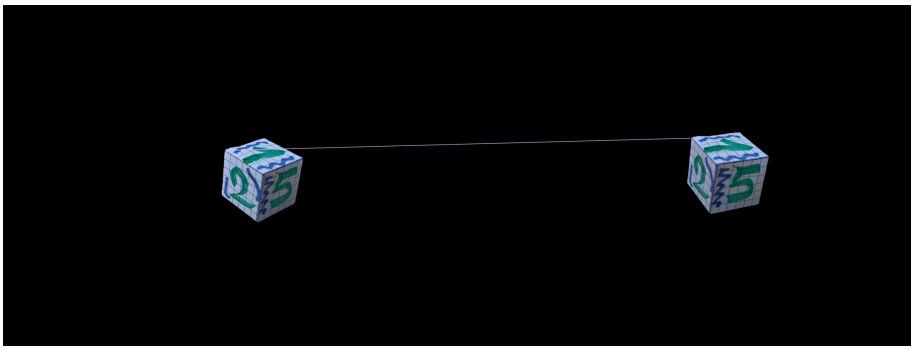

FIGURE 8 – Exemple d'une correspondance de points d'intérêt entre deux images (cercles violets liés par un trait)

3.5 Triangulation des points 3D

La triangulation des points est ensuite faite en trois étapes :

1. **L'estimation des matrices R_i et t_i** de rotation et translation entre les deux images i et $i+1$ à partir de la matrice essentielle E. Pour cela, nous avons utilisé la fonction `recoverPose` de OpenCV, qui permet de récupérer la rotation et la translation entre les deux images à partir de la matrice essentielle E et des points d'intérêt correspondants.

2. **La construction de la matrice de projection** P_i pour chaque image i à partir de la matrice intrinsèque K et des matrices de rotation et translation R_i et t_i. Cette matrice est faite après la combinaison de la rotation et de la translation estimées précédemment avec les R et t de toutes les images précédentes.
3. **La triangulation des points 3D** à partir des points d'intérêt correspondants et des matrices de projection P_i et P_{i+1}. Pour cela, nous avons utilisé la fonction `triangulatePoints` de OpenCV, qui permet de trianguler les points 3D à partir des points d'intérêt correspondants et des matrices de projection. Il est ici important de donner en paramètre les correspondances des points dans le bon ordre, c'est-à-dire que le point d'intérêt N de l'image i doit correspondre au point d'intérêt N de l'image $i + 1$.

3.6 Problèmes rencontrés

Durant le projet, nous avons rencontré plusieurs problèmes :

— La détection des points d'intérêt donnait parfois peu de points, voire aucun, ce qui rendait la triangulation impossible. Nous avons dû utiliser des objets 3D avec des motifs bien marqués pour que la détection fonctionne correctement.

— La correspondance a également posé problème lors de l'utilisation d'objets à motifs texturés périodiques. Dans ces cas, les points d'intérêt détectés pouvaient ne pas correspondre correctement entre les images, ce qui compliquait la reconstruction 3D. Nous avons donc dû utiliser des objets à texture unique (comme le cube de la figure 6) pour garantir une correspondance correcte des points d'intérêt.

— L'ordre des correspondances dans la liste se retrouvait auparavant dans un ordre aléatoire, sans garantir la cohérence point à point entre les images. Nous avons dû veiller à ce que les points d'intérêt soient correctement associés entre les images pour la triangulation.

— La fonction `recoverPose` de OpenCV, qui donne le vecteur translation *normalisé* et non-pas le vecteur translation réel nous a empêché de reconstruire la scène 3D correctement. L'application d'un facteur de mise à l'échelle aurait été possible en stéréoscopie, ou du moins en connaissant le déplacement réel de la caméra entre les deux images. Cependant, nous n'avions pas cette information, ce qui a rendu la reconstruction 3D à N images (plus de 2) impossible, on peut le constater dans la figure illustrant la trajectoire caméra estimée (figure 9).

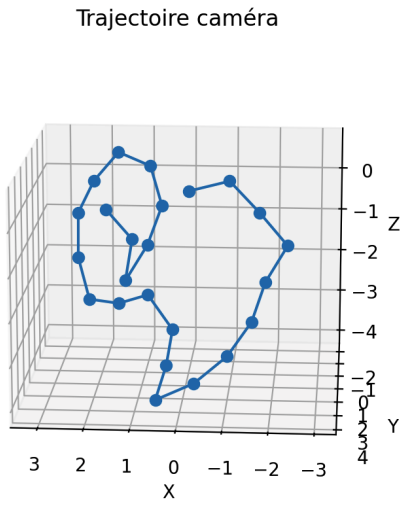

FIGURE 9 – Illustration de la trajectoire de la caméra estimée à partir des résultats de la fonction `recoverPose`

4 Résultats obtenus

Nous avons dans ce projet réussi à déterminer toutes les étapes de la reconstruction 3D d'une scène à partir d'images 2D, en utilisant la méthode de calibration de Zhang pour estimer la matrice intrinsèque K de la caméra. Cependant, nous n'avons pas réussi à obtenir une reconstruction 3D correcte de la scène à N images en raison des problèmes rencontrés lors de la triangulation des points 3D (évoqués dans la section précédente).
Nous avons donc décidé d'utiliser notre script avec seulement deux images pour observer les résultats de la projection des points. Nous avons observé les résultats suivants, où on distingue clairement deux plans imagés (figure 10 et figure 11).

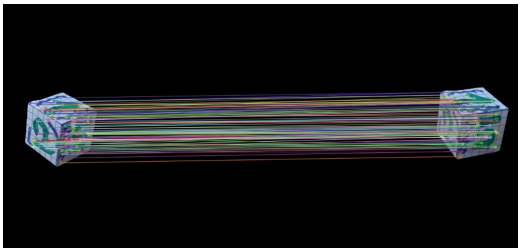

FIGURE 10 – Illustration des correspondances de points d'intérêt entre deux images

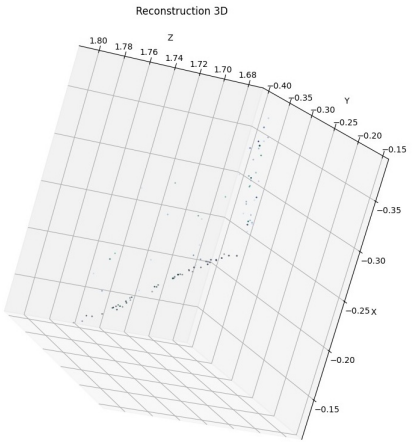

FIGURE 11 – Illustration de la reconstruction 3D à partir de deux images

Pour approfondir la vérification, nous avons sélectionné manuellement les points d'intérêt entre deux images, en dessinant le contour du cube. Les résultats sont visibles dans la figure 12 et la figure 13.

FIGURE 12 – Sélection des points d'intérêt manuellement entre deux images

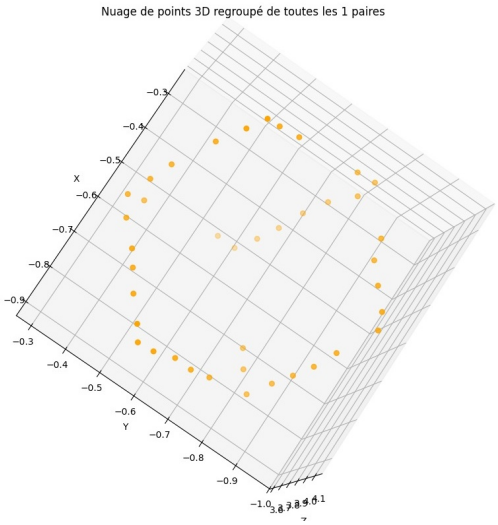

FIGURE 13 – Reconstruction en nuage de points sélectionnés manuellement

Nous avons fini cette étape de reconstruction en créant une texture à l'aide du nuage de points. Il est ici à noter que seulement 3 faces du cube sont visibles, ce qui donne la moitié d'un cube en texture (figure 14 et figure 15).

FIGURE 14 – Reconstruction du cube (face avant)

FIGURE 15 – Reconstruction du cube (face arrière)

Pour pousser ce projet plus loin (reconstruction à N images), l'utilisation d'une caméra stéréoscopique aurait été bénéfique, car elle aurait permis de connaître la

distance réelle entre les deux images et donc de reconstruire la scène 3D correctement. Une autre alternative aurait été un support rotatif gradué, permettant de connaître les rotations et translations caméra.

5 Où trouver notre travail ?

Tout le travail dont il est question dans ce rapport est disponible sur github.

Bibliographie

[1] Z. Zhang et al. A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, 1998. Consulté le 27 mai 2025.

[2] FrenchiDrone. Calibration de caméra. `https://www.frenchidrone.com/calibration-de-camera/`. Consulté le 20 mai 2025.

[3] Wikipédia. Calibration de caméra. `https://fr.wikipedia.org/wiki/Calibration_de_cam%C3%A9ra`. Consulté le 20 mai 2025.