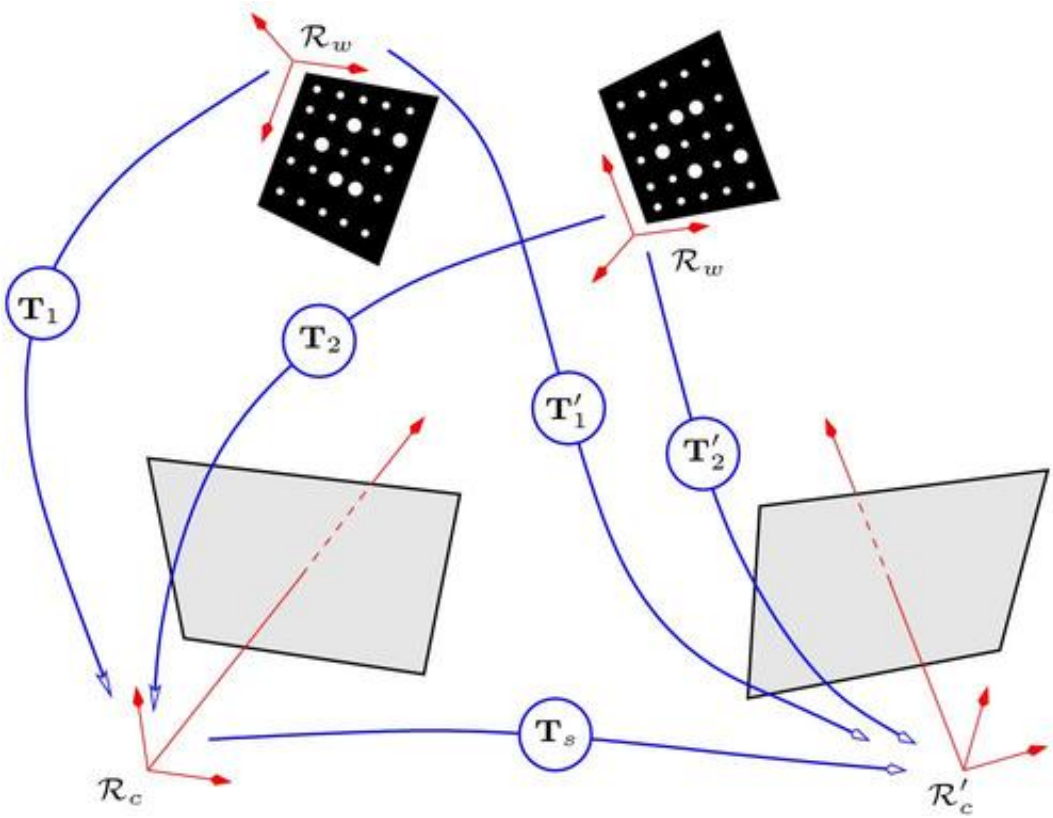
Calibrage d'une caméra

5 ETI – CPE Lyon

Majeure Image, modélisation et informatique

Noms, Prénoms : Simon Jeffrey, Rakotoarivony Aurore

Date : 04/10/2023



©https ://learnopengl.com/Getting-started/Hello-Triangle

1 Introduction

Dans le cadre de ce TP, notre objectif est de réaliser le **calibrage de la webcam Logitech C270**. Ce processus est essentiel pour permettre l'acquisition d'images en 2D à partir d'un environnement 3D continu. Nous devons projeter des objets de l'espace tridimensionnel sur un environnement bidimensionnel discrétisé en pixels, c'est-à-dire une image.
Ici, nous considérons la projection de l'objet sur l'image comme linéaire. Cependant, en pratique, il peut y avoir des distorsions optiques qui entraînent une déformation de l'espace image. Cette déformation peut être modélisée par des variations non linéaires, ce qui complexifie grandement le problème.
Nous utilisons une modélisation mathématique de la projection linéaire à l'aide de paramètres spécifiques. La connaissance de ces paramètres de projection a plusieurs utilités, que ce soit pour projeter de manière précise un objet modélisé sur une image, dans le cas de la réalité augmentée, ou pour pouvoir se repérer dans l'espace 3D à partir de l'image.
Il est donc important de comprendre le fonctionnement de la projection utilisée par une caméra. Les paramètres de cette projection peuvent être déterminés à partir de l'image capturée et des dimensions physiques de l'objet visible dans l'image. Pour ce faire, nous devons résoudre un problème inverse, nécessitant une reformulation du problème et une adaptation des paramètres.

Dans ce TP, nous utiliserons la **méthode de Tsai** pour atteindre cet objectif.

2 Projection

Pour effectuer une projection de l'objet sur l'image, nous utilisons deux types de paramètres : les paramètres extrinsèques et les paramètres intrinsèques.

Les **paramètres extrinsèques** sont liés au changement de repère entre l'objet 3D et la caméra. Ces paramètres décrivent la position et l'orientation de la caméra par rapport à l'objet dans l'espace 3D. Ils sont appelés extrinsèques car ils sont extérieurs à la caméra et dépendent de la configuration de la scène. Ces paramètres permettent de déterminer la transformation spatiale entre le système de coordonnées de l'objet et celui de la caméra.

Les **paramètres intrinsèques** sont liés à la caméra elle-même et au changement de repère de la caméra vers le repère de l'image. Ces paramètres incluent des informations sur les caractéristiques optiques de la caméra, telles que la distance focale et les facteurs d'échelle. Ils sont essentiels pour convertir les coordonnées 3D en coordonnées 2D sur l'image.

Soit u les coordonnées pixels de l'objet dans l'espace image et x ses coordonnées dans l'espace objet, nous voulons obtenir M la matrice de projection décomposée entre les matrices des projections intrinsèque et extrinsèque telles que :

$$u = Mx = M_{int}M_{ext}x$$

2.1 paramètres extrinsèques

Ces paramètres indiquent la position de la caméra par rapport au repère objet. Il est donc possible de les modéliser en les séparant en rotations et translations.
On souhaite trouver les coordonnées de l'objet, connues dans le repère objet $(x_1^o, x_2^o, x_3^o)^T$, dans le repère caméra $(x_1^c, x_2^c, x_3^c)^T$.

Soit :
$R_o(O, e_1^o, e_2^o, e_3^o)$ le repère objet centré autour de l'objet
$R_c(C, e_1^c, e_2^c, e_3^c)$ le repère caméra centré autour de la caméra.

Les **rotations** élémentaires sont fournies par les angles d'Euler (ϕ, γ, ω) qui représentent les rotations sur chaque composante du repère.
On peut obtenir une **translation** avec le centre du repère objet dans le repère caméra (o_1^c, o_2^c, o_3^c).

On obtient donc :

$$M_{ext} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & o_1^c \\ r_{21} & r_{22} & r_{23} & o_2^c \\ r_{31} & r_{32} & r_{33} & o_3^c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La quatrième dimension a été introduite afin de gérer la translation de manière linéaire.

2.2 paramètre intrinsèque

Une fois que nous avons trouvé les coordonnées de l'objet par rapport à la caméra, nous voulons trouver leur correspondance sur chaque pixel d'une image discrète 2D. Cela revient à exprimer les coordonnées de l'objet dans le repère image $(x_1^i, x_2^i, x_3^i)^T$ à partir de celles dans le repère caméra $(x_1^c, x_2^c, x_3^c)^T$. Nous conservons pour l'instant la troisième composante x_3^i, bien que nous nous intéressions aux deux premières, pour garder l'information de profondeur.
La profondeur est proportionnelle à x_3^c. Soit α le facteur de proportionnalité. Par le théorème de Thalès, ce dernier dépend de la distance entre l'image et la caméra, c'est-à-dire la focale. Nous avons ainsi :

$$\begin{pmatrix} \alpha x_1^i \\ \alpha x_2^i \\ \alpha x_3^i \\ \alpha \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1^c \\ x_2^c \\ x_3^c \\ 1 \end{pmatrix}$$

Nous avons désormais exprimé la relation entre la 3D et la 2D pour la profondeur. Il nous faut cependant enlever la troisième composante afin d'exprimer les grandeurs en pixels et non plus en millimètres.
Pour cela, nous allons utiliser les **facteurs d'échelle** (s_1, s_2) en $px.m^{-1}$ qui correspondent à la dimension d'un pixel. Nous allons également translater l'objet pour l'adapter au repère fixé par l'image. Ainsi, nous avons :

$$\begin{pmatrix} u_1 \\ u_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/s_1 & 0 & 0 & i_1 \\ 0 & 1/s_2 & 0 & i_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1^i \\ x_2^i \\ x_3^i \\ 1 \end{pmatrix}$$

Donc,

$$\begin{pmatrix} \alpha u_1 \\ \alpha u_2 \\ \alpha \end{pmatrix} = \begin{pmatrix} f/s_1 & 0 & 0 & i_1 \\ 0 & f/s_2 & 0 & i_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1^c \\ x_2^c \\ x_3^c \\ 1 \end{pmatrix}$$

Et donc

$$M_{int} = \begin{pmatrix} f/s_1 & 0 & 0 & i_1 \\ 0 & f/s_2 & 0 & i_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3 Problème inverse avec méthode de Tsai

Une fois les paramètres intrinsèques et extrinsèques de la projection trouvés et ayant connaissance de la valeur de la focale, nous pouvons appliquer la **méthode de Tsai**.

Elle consiste à poser un système linéaire mettant en lien les paramètres de la projection avec la connaissance des équivalents coordonnées objets et coordonnées pixels. Nous avons donc un ensemble de pixels $u = (\alpha u_1, \alpha u_2, \alpha)$ et nous pouvons retrouver leurs coordonnées en cm sur l'objet réel $x^o = (x_1^{o^k}, x_2^{o^k}, x_3^{o^k}, 1)$.
En posant $f_1 = \dfrac{f}{s_1}$ et $f_2 = \dfrac{f}{s_2}$, nous obtenons :

$$\begin{pmatrix} x_3^c u_1 \\ x_3^c u_2 \\ x_3^c \end{pmatrix} = \begin{pmatrix} f_1(r_{11}x_1^o + r_{12}x_2^o + r_{13}x_3^o + o_1^c) + i_1(r_{31}x_1^o + r_{32}x_2^o + r_{33}x_3^o + o_3^c) \\ f_2(r_{21}x_1^o + r_{22}x_2^o + r_{23}x_3^o + o_1^c) + i_2(r_{31}x_1^o + r_{32}x_2^o + r_{33}x_3^o + o_3^c) \\ r_{31}x_1^o + r_{32}x_2^o + r_{33}x_3^o + o_3^c \end{pmatrix}$$

En remplaçant x_3^c par son expression de la troisième ligne et en divisant, on obtient les deux composantes

$$\begin{pmatrix}\tilde{u}_1\\ \tilde{u}_2\end{pmatrix} = \begin{pmatrix}u_1 - i_1\\ u_2 - i_2\end{pmatrix} = \begin{pmatrix}f_1\dfrac{r_{11}x_1^o + r_{12}x_2^o + r_{13}x_3^o + o_1^c}{r_{31}x_1^o + r_{32}x_2^o + r_{33}x_3^o + o_3^c}\\ f_2\dfrac{r_{21}x_1^o + r_{22}x_2^o + r_{23}x_3^o + o_1^c}{r_{31}x_1^o + r_{32}x_2^o + r_{33}x_3^o + o_3^c}\end{pmatrix}$$

Le dénominateur étant le même, les composantes sont dépendantes et peuvent être assemblées :

$$\tilde{u}_1 f_2(r_{21}x_1^o + r_{22}x_2^o + r_{23}x_3^o + o_1^c) = \tilde{u}_2 f_1(r_{11}x_1^o + r_{12}x_2^o + r_{13}x_3^o + o_1^c)$$

. En posant $\beta = \dfrac{f_1}{f_2}$ et en divisant la ligne précédente par o_2^c des deux côtés, on obtient :

$$\tilde{u_1} = \left(\tilde{u_2}x_1^o, \tilde{u_2}x_2^o, \tilde{u_2}x_3^o, -\tilde{u_1}x_1^o, -\tilde{u_1}x_2^o, -\tilde{u_1}x_3^o\right)\begin{pmatrix}\dfrac{\beta r_{11}}{o_2^c}\\ \dfrac{\beta r_{12}}{o_2^c}\\ \dfrac{\beta r_{13}}{o_2^c}\\ \dfrac{r_{21}}{o_2^c}\\ \dfrac{r_{22}}{o_2^c}\\ \dfrac{r_{23}}{o_2^c}\end{pmatrix}$$

En prenant plusieurs points et pixels, l'objectif est accompli.
Trouver un système linéaire $AL = U_1$ avec A et U_1 les coefficients connus des points et pixels, et L un vecteur contenant les paramètres de la projection à estimer.
Cette estimation se fait grâce à l'erreur quadratique au sens des moindres carrés. Nous prenons suffisamment de pixels pour que le problème soit surdéterminé (plus d'équations que de paramètres à estimer), nous pouvons donc estimer L avec une équation normale impliquant la pseudo-inverse de A. Nous pouvons ainsi retrouver les paramètres de L. Nous devons être prudents concernant une chose, le signe de o_2^c, car son expression est en valeur absolue :

$$|o_2^c| = \frac{1}{\sqrt{l_5^2 + l_6^2 + l_7^2}}$$

Nous posons l_i le coefficient de la i^{me} ligne de L. Ce coefficient dépend évidemment de l'orientation du repère caméra par rapport au repère objet.
Nous avons donc une expression de o_2^c, β, o_1^c et des vecteurs rotation sur le premier et deuxième axe. Dans notre simplification du problème, nous n'avons pas estimé le 3ème axe r_3 qui peut être obtenu par le produit vectoriel entre r_1 et r_2, les rotations étant des isomorphismes choisies dans un repère orthonormé. Nous en déduisons les angles d'Euler.
Dans la simplification du problème, nous avons également regroupé f_1 et f_2 dans un seul paramètre β et nous avons simplifié des termes en enlevant l'information sur o_3^c. Nous devons donc les estimer. Nous savons néanmoins en utilisant les équations précédentes que

$$\tilde{u}_2 o_c^3 = -f_2(r_{21}x_1^o + r_{21}x_2^o + r_{23}x_3^o) = -\tilde{u}_2(r_{31}x_1^o + r_{31}x_2^o + r_{33}x_3^o)$$

Ainsi, nous obtenons le nouveau système linéaire

$$B\begin{pmatrix}o_3^c\\ f_2\end{pmatrix} = R$$

Nous déduisons f_1, s_1,s_2 de la focale, de β, et de f_2. Nous avons donc tous les paramètres du calibrage.

4 Acquisition de la Mire

Pour réaliser le calibrage, nous utilisons une mire. En effet, la mire est une figure géométrique simple, sur laquelle on peut très facilement associer une liste de pixels correspondant à l'intersection de

cases. L'intersection peut être trouvée grâce à un algorithme de points d'intérêt, par exemple. La mire est composée de 11×8 carrés de 20 mm. On peut aisément trouver la position géométrique des intersections. Deux photos seront utilisées car nous devons placer la mire à une hauteur différente (ici 100 mm) afin d'avoir la profondeur.

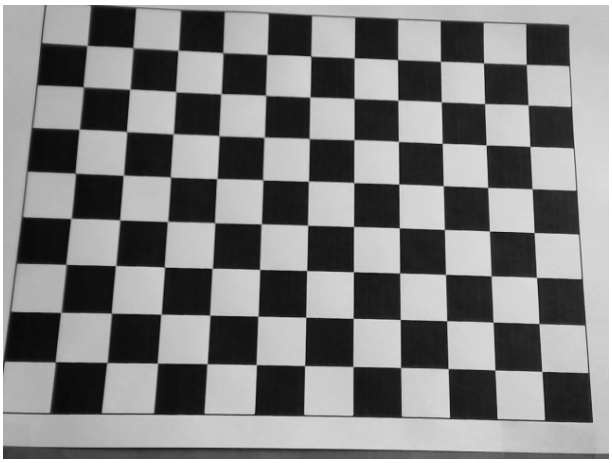


FIGURE 1 – Mire à 0 mm

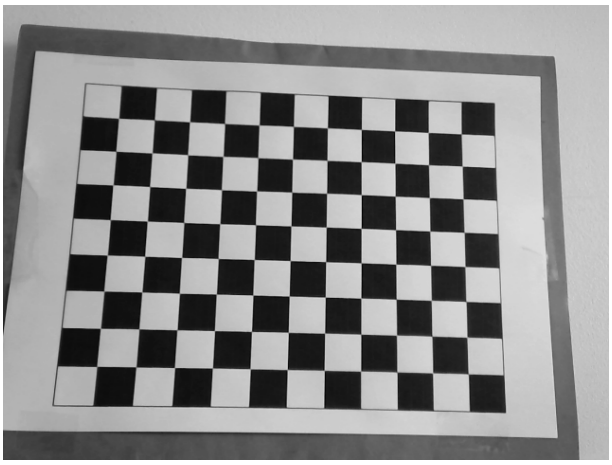


FIGURE 2 – Mire à 100 mm

Pour effectuer le calibrage, nous devons associer les coordonnées pixels de l'objet avec les coordonnées réelles en millimètres. La première étape consiste donc à trouver les coordonnées en pixels des coins. Une fonction est dédiée à cela en Python.
Nous avons maintenant un ensemble de coordonnées en pixels.
La deuxième étape consiste à associer à ces coordonnées en pixels un ensemble bijectif de coordonnées en millimètres. Nous pouvons poser le repère de l'objet sur la première intersection. Nous pouvons exprimer toutes les intersections à partir de la première, qui sera placée à la coordonnée $(0, 0, 0)$, car nous savons qu'elles sont espacées de 20 mm et que la deuxième image aura un décalage de 100 mm en hauteur. La difficulté consiste à associer dans le bon ordre les coordonnées en millimètres avec les coordonnées en pixels. En effet, OpenCV n'a pas choisi le même repère pour les coordonnées pixels. Donc, si nous voulons les bonnes associations, notre ensemble de coordonnées en millimètres ne doit pas commencer à son origine. Voici un schéma explicatif de ce que nous avons.

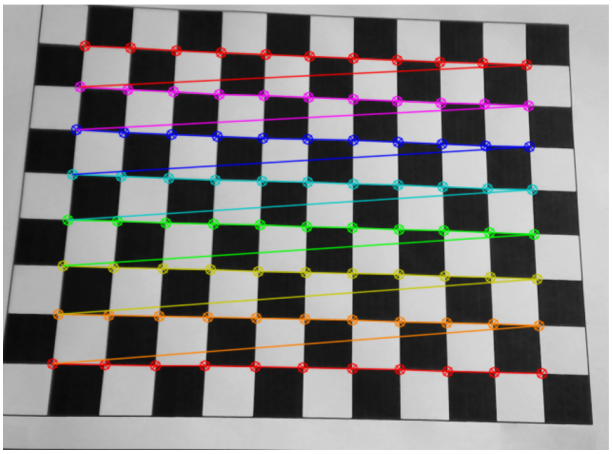


FIGURE 3 – Coins de la mire à 0 mm

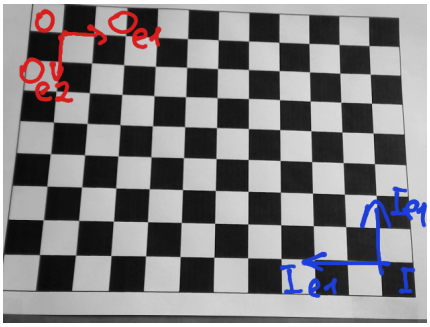


FIGURE 4 – En bleu le repère image en rouge le repere objet

5 Calibrage de la Caméra

La première étape consiste à poser le système linéaire décrit précédemment : $AL = U_1$. Nous cherchons la matrice L contenant les informations sur les paramètres de projection. Les matrices A et U_1 sont construites à partir d'une combinaison des vecteurs de positions en pixels et en millimètres. Pour résoudre l'équation normale au sens des moindres carrés, nous devons trouver la matrice pseudo-inverse de A. Nous avons donc :

$$L = (A^T A)^{-1} A^T U_1$$

Ici, $(A^T A)^{-1} A^T$ est la pseudo-inverse de A, car A modélise un système surdéterminé, n'est donc pas carrée et inversible.
Les paramètres se déduisent des composantes de L. Un des paramètres, o_2^c, est néanmoins obtenu en valeur absolue. Son signe doit donc être réfléchi. o_2^c est la position de l'origine du repère objet (situé en haut à gauche arbitrairement) par rapport à la caméra. Or, nous avons supposé la caméra placée au milieu de l'image. En conclusion, le repère objet est situé en haut à gauche de la caméra, ce qui correspond à un o_1^c et o_2^c négatifs. o_3^c, obtenu ultérieurement, sera de signe positif, car sinon cela voudrait dire que l'objet sera placé derrière la caméra et ne sera pas visible.

Dans notre cas de figure, nous avons obtenu :

$$o_2^c = \frac{1}{l_5^2 + l_6^2 + l_7^2} = -117.68 \text{ mm}$$
$$\beta = |o_2^c|\sqrt{l_1^2 + l_2^2 + l_3^2} = 1.006$$
$$o_1^c = \frac{l_4 o_2^c}{\beta} = -63.48 \text{ mm}$$
$$r_{11} = l_1 \frac{o_2^c}{\beta} = 0.99$$
$$r_{12} = l_2 \frac{o_2^c}{\beta} = -0.04$$
$$r_{13} = l_3 \frac{o_2^c}{\beta} = 0.00$$
$$r_{21} = l_5 o_2^c = 0.02$$
$$r_{22} = l_6 o_2^c = 0.99$$
$$r_{23} = l_7 o_2^c = 0.09$$

Nous observons la cohérence des résultats. o_2^c et o_1^c sont bien de signes négatifs, avec des proportions cohérentes avec la taille de l'image. β, qui est le rapport des dimensions par pixel, est proche de 1, car même si les dimensions par pixels changent sur les deux axes, nous ne retrouvons pas des formes déformées. Il est plus aisé de vérifier les valeurs des vecteurs rotation grâce aux valeurs des angles d'Euler. Nous pouvons néanmoins vérifier qu'ils ont une norme de 1, ce qui est le cas. Nous pouvons trouver le vecteur r_3 par produit vectoriel car nous savons que ces vecteurs correspondent aux angles d'Euler et correspondent donc à un repère orthogonal.

$$\phi = -\arctan\frac{r_{23}}{r_{33}} = -0.09$$
$$\gamma = -\arctan\frac{r_{12}}{r_{11}} = -0.02$$
$$\omega = \arctan\frac{r_{12}}{-r_{23}\sin\phi + r_{33}\cos\phi} = 0$$

Nous retrouvons des angles en radians de petite valeur. En effet, nous avons essayé d'aligner la caméra par rapport à la mire, afin de conserver la droiture de celle-ci. Cela est donc cohérent.
Il nous reste donc à déterminer f_2 et o_3^c avec la méthode décrite précédemment avec des moindres carrés.

$$B\begin{pmatrix} o_3^c \\ f_2 \end{pmatrix} = R$$

$$o_3^c = 348.09 \text{ mm}$$
$$f_2 = 814.22$$

On obtient donc

$$f_1 = \beta f_2 = 819.59$$
$$s_1 = \frac{f}{f_1} = 0.005 \text{ mm/px}$$
$$s_2 = \frac{f}{f_2} = 0.005 \text{ mm/px}$$

Avec une image de dimension 480x640 px, on en déduit que la taille du capteur, à partir des dimensions d'un pixel, est de 2.34x3.14 mm.
En réutilisant les deux matrices de projections de la partie 2, nous pouvons projeter des coordonnées en millimètres sur l'image. Nous pouvons vérifier que nos coordonnées en millimètres correspondent bien aux intersections. Pour projeter, nous arrondissons les coordonnées en pixels à l'entier le plus proche.

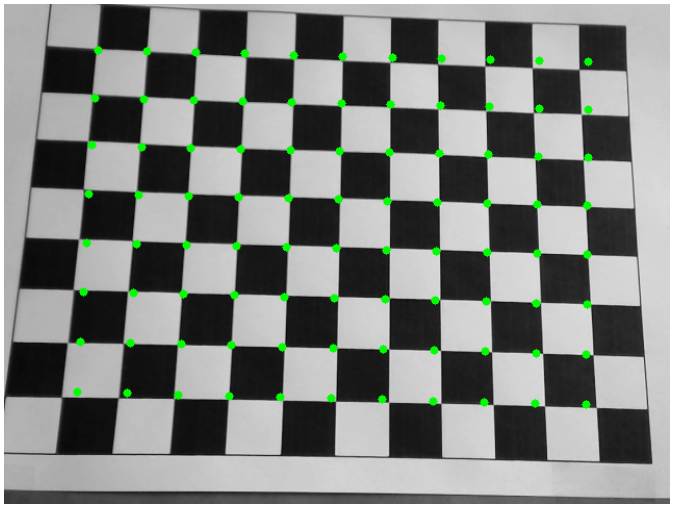


FIGURE 5 – Résultat de la projection

Nous obtenons un résultat plutôt bon. Cependant, nous pouvons constater la présence de distorsions optiques car sur les extrémités, nous avons une légère erreur. Soit u les coordonnées pixels acquises considérées comme vérité terrain, et $\hat{u}$ les coordonnées pixels estimées à partir de la projection. On calcule l'erreur

$$\epsilon = \frac{\sum_{i=1}^{n} ||\hat{u}_i - u_i||^2}{n}$$

Nous obtenons une déviation ϵ d'environ 4 px.

6 Comparaison des résultats avec `cv2.calibrateCamera`

Nous n'avons pas réussi à utiliser la fonction `cv2.calibrateCamera`. En effet, lorsque nous utilisons des données non coplanaires (la hauteur change), la fonction demande l'utilisation d'une estimation de la projection intrinsèque. Nous lui avons fourni cette projection, mais l'erreur de `cv2` persistait comme si on ne lui avait rien donné. Néanmoins, nous avons réussi à la faire fonctionner pour des données coplanaires avec une seule image, mais ce n'était pas ce que nous voulions.

7 Conclusion

Ce TP de calibrage de webcam en utilisant la méthode de Tsai nous a permis de trouver une matrice de projection afin d'associer le monde réel aux coordonnées pixels de l'image. Nous avons rencontré quelques soucis avec la fonction `calibrateCamera`, donc nous n'avons pas pu comparer avec des méthodes de corrections de distorsions optiques non linéaires.