

# TP vision 3D

## Option SMR, ENSMM

Antoine ANDRE

2020 - 2021

## Introduction

La vision par ordinateur constitue un moyen précieux pour effectuer des mesures sur des objets sans contact. Même si l'utilisation d'une seule caméra permet d'obtenir bon nombre d'informations, cette méthode se retrouve rapidement limitée lorsque des tâches plus complexes comme l'estimation de pose ou la reconstruction d'objets est considérée.

Cette série de travaux pratique a pour but de mettre en avant quelques techniques utilisées en vision par ordinateur en considérant plusieurs caméras. Ceux-ci ont pour but d'aboutir à la reconstruction d'objets imagés via deux caméras en utilisant la triangulation 3D.

Cette tâche complexe nécessite cependant de maîtriser plusieurs notions qui constitueront l'ensemble des TPs. La première séance sera dédiée à l'étalonnage de caméra qui permettra d'obtenir les paramètres intrinsèque des caméras utilisées lors des TPs. La seconde séance s'intéressera à l'homographie en considérant plusieurs cas d'études comme la mosaïque ou encore la rectification d'images. Enfin la dernière séance sera consacrée à l'étude de la triangulation avec deux caméras et qui utilisera les notions acquises lors des deux séances précédentes. A l'issue de cette série de travaux pratiques, il sera possible de reconstruire plusieurs faces d'un même objet observé.

Afin d'étendre la portée de ces travaux pratiques, plusieurs cas concrets d'études sont proposés dans les sections "Pour aller plus loin...". Leurs but est d'apporter une vision plus globale et pratique sur la vision 3D par ordinateur en étudiant des alternatives aux solutions proposées lors des TPs et en montrant plusieurs cas courant d'utilisation de la vision 3D.

A l'issue de ces travaux pratiques, les étudiants doivent fournir un compte rendu soit sous la forme d'un document computationnel (Jupyter notebook avec python par exemple) qui devra contenir le code exécutable ainsi que les réponses aux différentes questions des travaux pratiques. Les étudiants peuvent également travailler avec Matlab et devront rendre les sources de leurs projets ainsi qu'un document annexe contenant les réponses aux questions des différents travaux pratiques.

# 1 TP1 : étalonnage de caméra

## 1.1 Objectifs du TP

L'étalonnage de caméra est la base de tout processus d'étude de système de vision par ordinateur, en effet il est primordial de connaître les paramètres d'une caméra pour pouvoir passer de points réels (3D) vers les points imaginés sur la caméra (2D) (et réciproquement).

Ainsi le but de ce TP sera d'effectuer un étalonnage de caméra qui sera réutilisé plus tard pour le dernier TP. Il faut donc bien veiller à ce que les résultats obtenus lors de cette séance soient affinés autant que possible et qu'ils soient également sauvegardés pour ne pas avoir à refaire un étalonnage lors des prochaines séances.

## 1.2 Rappels mathématiques

On rappelle que pour tout point de l'espace correspond un point image calculé à l'aide de l'équation suivante:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

Qui peut être écrite sous la forme suivante :

$${}^i\tilde{p} \equiv K (I O) {}^cT_o {}^o\tilde{P} \quad (2)$$

**Question 1 :** Rappelez quel est le but de l'étalonnage de caméra ainsi que ses étapes de résolution.

**Question 2 :** La matrice  $K$  présentée dans l'équation 1 n'est pas complète, que manque-t-il pour que le modèle de projection soit complet ?

**Question 3 :** Avant d'effectuer l'étalonnage de la caméra, listez les différentes applications concrètes dans les cas où :

- $K$ ,  ${}^cT_o$  et  ${}^o\tilde{P}$  sont connus
- ${}^i\tilde{p}$  et  ${}^o\tilde{P}$  sont connus
- ${}^i\tilde{p}$ ,  $K$  et  ${}^cT_o$  sont connus

## 1.3 Travail à réaliser

### 1.3.1 Utilisation de l'outil Jupyter

Tout comme pour le TDAO fait plus tôt, le but est d'étalonner une caméra. Cette fois-ci nous utiliserons la librairie **OpenCV**, couplée à l'interpréteur python **Jupyter**.

Pour ceux qui souhaitent utiliser **Matlab** comme logiciel de base pour les TPs, il est conseillé d'utiliser soit la toolbox de Bouguet (`calib_toolbox` : [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)). Dans ce cas, veuillez vous référer au TDAO effectué plus tôt pour l'étalonnage de caméra. Une autre solution est d'utiliser directement la toolbox Matlab de vision par ordinateur qui fournit une interface d'étalonnage de caméra.

Ouvrez un nouveau projet **Jupyter notebook** (ou Jupyter lab) et initialisez votre document computationnel avec les librairies suivante :

numpy	manipulations mathématiques
opencv	manipulations d'images
glob	manipulation de fichiers
matplotlib (pyplot et mpimg)	affichages
math (cos, sin et $\pi$ )	fonctions mathématiques usuelles

Ci-dessous la cellule d'initialisation du notebook Jupyter:

```
import numpy as np
import cv2
import glob
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import math
from math import cos, sin, pi
```

### 1.3.2 Mise en place de la caméra

En fonction du matériel à disposition, utilisez soit une caméra du poste de travail (IDS uEye), soit directement votre smartphone pour effectuer l'acquisition des mires.

Si vous utilisez les caméras du poste de travail, allumez les lampes de la table éclairante et lancez le logiciel **uEye** en choisissant une acquisition monochrome. Le mires à utiliser pour l'étalonnage sont déjà placées sur le poste de travail.

Si vous utilisez votre téléphone portable pour étalonner sa caméra et que vous ne possédez pas de mire, il est possible d'effectuer l'étalonnage sans utiliser d'imprimante. Pour cela, rendez vous sur le site [https://www.mrpt.org/downloads/camera-calibration-checker-board\\_9x7.pdf](https://www.mrpt.org/downloads/camera-calibration-checker-board_9x7.pdf).

Dans le premier cas, l'acquisition des images de l'échiquier se font avec la caméra fixe et une nouvelle pose de l'échiquier pour chaque image. Dans le second cas, c'est l'échiquier qui est fixe et la caméra qui prend une nouvelle pose à chaque acquisition.

Afin d'effectuer un étalonnage aussi précis que possible tout en évitant un coût en temps trop important lors du traitement des images, il faut choisir le nombre d'acquisitions à effectuer.

**Question 4 :** Combien d'images allez-vous acquérir pour effectuer l'étalonnage de la caméra ?

**Remarque :** Les images doivent être placées de manière rigoureuse dans un dossier à part et doivent porter un nom générique compréhensible par le logiciel (type "img\_1.png").

### 1.3.3 Etalonnage de la caméra à l'aide de la librairie OpenCV

Avant d'effectuer l'étalonnage de la caméra, il faut penser à la forme qu'auront les images à traiter. Comme les images prises avec des téléphones portables sont souvent volumineuses, créez une fonction python qui permet de réduire d'un facteur d'échelle la taille de l'image acquise (afin de diminuer le temps de calcul).

```
def resize_img(img, scale):
    pass
    # Réalisez ici la réduction de l'image
    # Utilisez la bibliothèque OpenCV
    # Retournez l'image réduite
    return img
```

**Question 5 :** Implémentez la fonction `resize_img()`.

Pour effectuer l'étalonnage de la caméra, consultez la documentation d'OpenCV qui fournit un exemple d'étalonnage à l'url suivante : [https://docs.opencv.org/master/dc/dbb/tutorial\\_py\\_calibration.html](https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html).

**Question 6 :** Effectuez l'étalonnage des images acquises.

**Question 7 :** Etudiez l'erreur de reprojection de l'étalonnage de la caméra.

**Question 8 :** Sauvegardez les résultats des paramètres intrinsèques de la caméra en utilisant la fonction `numpy.savez()`.

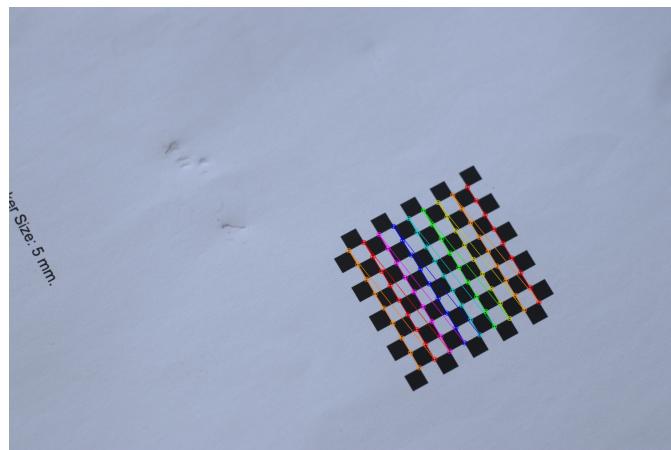


Figure 1: Résultat de l'étalonnage d'une caméra avec les intersections de l'échiquier reprojetés

## 2 TP2 : homographie

### 2.1 Objectifs du TP

Une application courante de la vision par ordinateur est d'effectuer la correspondance entre deux vues qui sont dans le même plan, c'est ce qui est appelé une homographie. Le but de ce TP sera de mettre d'étudier comment mettre en place une fonction permettant d'effectuer un tel calcul.

Ce TP sera donc organisé en plusieurs sections qui comprendront l'étude théorique du calcul d'homographie, la définition de fonction de calcul de la matrice  $H$  de l'homographie et l'application de cette fonction à plusieurs cas de figures.

### 2.2 Rappels mathématiques

On rappelle qu'une homographie est la transformation d'un plan projectif en un autre. Ainsi, un point  $p_2$  d'une image  $I_2$  est obtenu par l'application de l'homographie  ${}^2H_1$  au point  $p_1$  de l'image  $I_1$  :

$$\tilde{p}_2 \approx {}^2H_1\tilde{p}_1 \quad (1)$$

où la notation  $\sim$  désigne les coordonnées homogènes d'un point et  $\approx$  désigne l'égalité à un facteur d'échelle près (ou, dit autrement, la proportionnalité).

Pour se débaraqqer de cette égalité projective, une astuce est d'utiliser le produit vectoriel :

$$\tilde{p}_2 \wedge ({}^2H_1\tilde{p}_1) = 0_{3 \times 1} \quad (2)$$

ou encore, à l'aide de la matrice antisymétrique de pré-produit vectoriel associé au vecteur  $\tilde{p}_2$ , que l'on note  $[\tilde{p}_2]_\wedge$  :

$$[\tilde{p}_2]_\wedge {}^2H_1\tilde{p}_1 = 0_{3 \times 1} \quad (3)$$

qui après manipulation algébrique se réécrit :

$$([\tilde{p}_2]_\wedge \otimes {}^t\tilde{p}_1) h = 0_{3 \times 1} \quad (4)$$

où  $\otimes$  désigne le produit de Kronecker et

$$h = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \text{ tel que } H = \begin{pmatrix} {}^t h_1 \\ {}^t h_2 \\ {}^t h_3 \end{pmatrix} \quad (5)$$

On peut donc calculer l'homographie  $H$  à partir de 4 en ayant pris soin de normaliser les coordonnées des points dans  $[-1, 1] \times [-1, 1]$  à l'aide d'une matrice  $K$  de la forme suivante qui sera à définir :

$$K = \begin{pmatrix} \alpha & 0 & u \\ 0 & \beta & v \\ 0 & 0 & 1 \end{pmatrix} \quad (6)$$

### 2.3 Travail à réaliser

Dans un premier temps, l'homographie étudiée sera basée sur deux images des quais de Grenoble qui à terme formeront une seule image appelée moasique (ou panorama).

**Remarque :** cette partie du TP peut s'effectuer sans l'utilisation de la librairie OpenCV. Les librairies numpy et matplotlib suffisent en effet pour calculer l'homographie.

### 2.3.1 Extraction de points d'une image

Après avoir importé les deux images supports de l'homographie, créez deux tableaux contenant les points en correspondance qui permettront de calculer l'homographie.

**Question 1 :** Combien de points en correspondance faut-il sélectionner ? Pourquoi ?

Afin de vérifier que les points correspondent bien à ceux sélectionnés, affichez les dans le document computationnel avec la librairie Matplotlib.

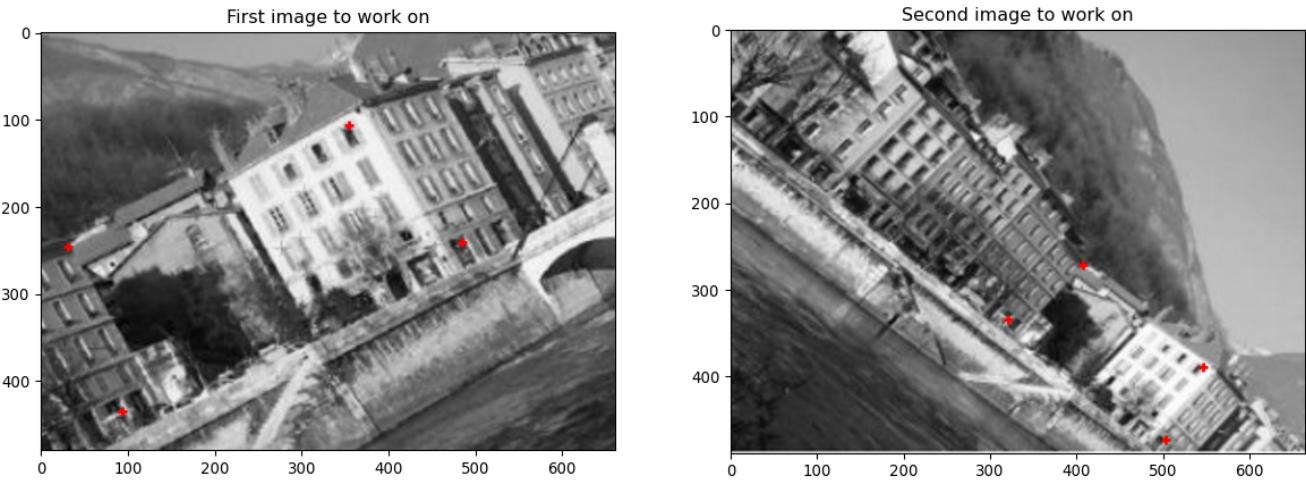


Figure 2: Exemple avec 4 points en correspondance dans chacune des images.

### 2.3.2 Normalisation des points

Afin de pouvoir calculer l'homographie entre les deux images, les points sélectionnés doivent se trouver dans le même espace normalisé (*i.e.* dans l'espace  $[-1, 1] \times [-1, 1]$ ). L'équation 6 nous renseigne sur la forme que doit avoir cette matrice.

**Question 2 :** Calculez les coefficients  $\alpha$ ,  $\beta$ ,  $u$  et  $v$  de la matrice  $K$  définie dans l'équation 6.

**Question 3 :** Implémentez la fonction `normalise_pts` qui prend en entrée un point de coordonnées homogènes de l'image et le normalise dans l'espace  $[-1, 1] \times [-1, 1]$ .

```
def normalise_pts(pts):  
  
    # Implementation of the function  
  
    return pts_h
```

### 2.3.3 Calcul de l'homographie

L'équation 4 nous indique directement comment obtenir une homographie entre deux séries de points mis en correspondance.

**Question 4 :** Quel calcul mathématique faut-il implémenter dans la fonction de calcul d'homographie ?

**Question 5 :** Implémentez la fonction `calcul_homographie(Pts1, Pts2)` qui prend en entrée deux séries de points en correspondance et retourne la matrice d'homographie associée.

```
def calcul_homographie(Pts1, Pts2):  
  
    # Calcul de la matrice d'homographie H  
  
    return H
```

**Question 6 :** Comment vérifier la validité de la matrice d'homographie calculée ?

### 2.3.4 Création d'une mosaïque de 2 images d'une scène plane

Pour créer une mosaïque de deux images d'une scène plane, il "suffit" de déterminer l'homographie entre les deux images puis de coller la transformée de l'une sur l'autre.

L'algorithme de base est le suivant :

1. Extraire (au moins) 4 points de chaque image
2. Calculer l'homographie  ${}^1H_2$
3. Afficher la couleur de chaque pixel correspondant de  $I_2$  dans  $I_1$

**Question 7 :** Implémentez cet algorithme et afficher le résultat avec  $I_2$  collé dans  $I_1$ , puis avec  $I_1$  collé dans  $I_2$ .

Bien que cet algorithme fonctionne bien, il présente deux défauts majeurs, l'un d'eux est (évidemment) le temps de calcul assez conséquent.

**Question 8 :** Qu'est-ce qui altère le résultat de la mosaïque, surtout lorsque l'image collée est plus grande que l'image originale ?

Pour pallier ce problème, un autre algorithme, décrit ci-dessous peut-être utilisé :

1. Extraire (au moins) 4 points de chaque image
2. Calculer l'homographie  ${}^1H_2$
3. Calculer la transformée des 4 coins de  $I_2$  dans  $I_1$
4. En déduire les dimensions de l'image résultat  $I_r$
5. Pour chaque pixel  $p_r$  de  $I_r$  :
  - (a) Si  $p_r$  appartient à  $I_1$ , stocker sa couleur
  - (b) Calculer l'antécédent  $p_2$  de  $p_r$  par  ${}^1H_2$
  - (c) Si  $p_2$  appartient à  $I_2$ , stocker sa couleur
  - (d) Déterminer la couleur finale de  $p_r$

**Question 9 :** Quelle est la différence entre les deux algorithmes ?

**Question 10 :** Implémentez ce deuxième algorithme et testez le.

### 2.3.5 Vers des mosaïques plus fines

Au cours de ce TP nous avons vu comment créer une mosaïque avec une image de taille modeste et en noir et blanc. De plus, la sélection des points était manuelle, ce qui rend la tâche difficile à automatiser. Enfin, le programme que nous avons créé ne fonctionne que lorsque les images sont en noir et blanc, ce qui ne reflète pas la réalité de la plupart des acquisitions qui sont en couleurs (sur 3 canaux).

**Question 11 :** Implémentez la fonction `mosaique(img1, pts1, img2, pts2)` qui utilise l'algorithme décrit dans la section précédente prenant en entrée deux images et deux séries de points en correspondance et renvoie la mosaïque des deux images.

Une fois cette fonction créée, il devient plus simple de faire de manière plus "automatisée" le calcul de mosaïque. Comme énoncé plus haut, la plupart des images acquises sont en couleur, en changeant quelques paramètres de la fonction `mosaique`, celle-ci peut répondre à ce critère. Afin de tester la méthode, effectuez l'acquisition de deux images de votre choix (ou de deux images déjà acquises propices à devenir une mosaïque).

**Question 12 :** Augmentez la fonction `mosaique()` pour qu'elle renvoie une image en couleur.

**Question 13 :** Testez votre programme sur deux images en correspondance acquises par vos soins.

## 2.4 Pour aller plus loin...

Cette section dérit des pistes d'exploration qui utilisent l'homographie comme base de calcul. Aucune de ces questions n'est primordiale pour la suite du TP, mais s'il vous reste du temps ou que vous souhaitez pousser la réflexion sur des cas de figures concrets, ces pistes sont là pour ça.

### 2.4.1 Une automatisation complète du calcul d'homographie ?

Au cours de ce TP nous avons pu effectuer plusieurs homographies avec des points en mis en correspondance manuellement. Cependant, d'autres techniques plus rapides existent. L'une d'elle repose sur un tirage aléatoire d'homographies et se nommre RANSAC (pour RANdom SAmples Consensus).

*Question 14 :* Décrivez le principe de fonctionnement d'un tel algorithme.

### 2.4.2 Rectification d'images

Une application courante de l'homographie consiste à effectuer des rectifications locales (ou globales) d'images pour en redresser une portion. Par exemple cette méthode est utilisée dans la plupart des applications pour smartphone visant à scanner des documents. Ainsi, même si le document n'est pas imaginé parfaitement dans le plan perpendiculaire à l'axe optique, une rectification peut être effectuée. En effet, les proportions de l'image de départ sont connues et il suffit de reporter ces proportions dans l'image d'arrivée qui sera par conséquent bien droite. Si une image d'un document A4 est considéré par exemple, il faudra créer une homographie entre les coins du document et les coins d'une image de même proportion que le A4.

De même, la plupart des radars de plaque qui sont difficiles à placer donnent des images de plaques minéralogiques en biais. Comme les proportions d'une plaque de voiture sont connues, il devient facile de rectifier une portion de l'image contenant une plaque et ainsi faciliter la reconnaissance de la plaque.

*Question 15 :* Effectuez une rectification d'une image choisie par vos soins.



Figure 3: Résultat de la rectification d'une partie d'une image (source : tableau de Chifaa).

#### 2.4.3 Incorporation d'image dans une scène

Il est également possible de faire l'inverse de la rectification, et donc d'intégrer des images droites dans des images non rectifiées. Une application courante se trouve dans les logiciels de traitement d'images et trouvent tout leur intérêt quand un suivi de mouvement est considéré. En effet, intégrer une scène dans une autre en admettant qu'elles sont dans le même plan est un effet largement utilisé.

*Question 16 :* Incrustez une image droite dans une autre en utilisant l'homographie.

Même si ces techniques semblent primitives, elles sont pourtant à la base de la plupart des traitements d'images 3D. En rajoutant une vitesse de calcul accrue ainsi qu'une sélection automatique des points de l'homographie il devient possible d'effectuer toutes sortes de tâches d'incrustation ou de rectification.

## 3 TP3 : triangulation

### 3.1 Objectifs du TP

La dernière séance de travaux pratiques s'intéresse à la reconstruction d'objet imagé avec deux caméras. Plusieurs méthodes existent et dépendent des paramètres connus à priori. Dans notre cas, nous considérons deux caméras imageant le même objet et dont la pose relative est connue. De fait, bien que la scène observée soit totalement inconnue, il va quand même être possible de reconstruire des parties de celle ci.

Pour ce faire nous utiliserons la triangulation 3D et nous nous intéresserons plus particulièrement à la reconstruction des faces d'un objet. Ce TP sera organisé en plusieurs sections qui comprendront l'étude théorique de la triangulation de caméras, la définition de plan 3D ainsi que la reconstruction en nuage de points des faces de l'objet observé.

### 3.2 Rappels mathématiques

Dans le cas de figure de la triangulation 3D, nous retirons l'hypothèse que la scène est plane. Tout comme pour l'homographie, les points de l'image de gauche ainsi que ceux de l'image de droite sont connus. La matrice de transformation du repère caméra 1 vers le repère caméra 2 est également connu.

L'image de gauche peut être décrite avec le modèle de projection suivant :

$$[{}^{img}\tilde{p}_g] {}^{img}K_g (I \ O) {}^cT_g {}^g\tilde{P} = 0 \quad (1)$$

Le repère objet est arbitrairement fixé sur le repère de gauche ( $R_o = R_g$ ). On a donc :

$$[{}^{img}\tilde{p}_g] \wedge ({}^{img}K_g \ 0) {}^g\tilde{P} = 0 \quad (2)$$

Pour la caméra de droite, on a :

$$[{}^{img}\tilde{p}_d] {}^{img}K_d (I \ O) {}^dT_g {}^g\tilde{P} = 0 \quad (3)$$

avec  ${}^dT_g$  la matrice de transformation permettant de passer de la caméra 1 vers la caméra 2. Ainsi nous pouvons extraire deux équations liées qui vont permettre de pouvoir remonter aux coordonnées du point  $P$ .

### 3.3 Travail à réaliser

#### 3.3.1 Acquisition de deux images en correspondance

Pour mener à bien la triangulation des points 3D observés à reconstruire, il faut effectuer deux acquisitions avec une pose connue entre les deux caméras. Pour ce faire, le plus simple et d'effectuer une rotation de l'objet observé autour d'un axe (l'axe  $y$  par exemple) d'une dizaine de degrés.

La matrice de transformation entre les deux caméras sera donc la combinaison d'une rotation simple ainsi que d'une translation calculable facilement grâce à la connaissance de l'angle effectué.

Le choix de l'objet à reconstruire doit présenter des faces assez planes pour que le processus mis en place lors de cette étude fonctionne correctement.

**Question 1 :** Effectuez l'acquisition de deux images d'un objet possédant des faces distinctes et avec une transformation entre les deux caméras facile à retrouver.

Les faces de l'objet observé seront reconstruits à partir de plans créés via 3 points triangulés. Ces points doivent être choisis avec soin et retranscrire avec efficacité le plan à reconstruire.

**Question 2 :** choisissez 3 points (ou plus) définissant un plan de l'objet observé à reconstruire.

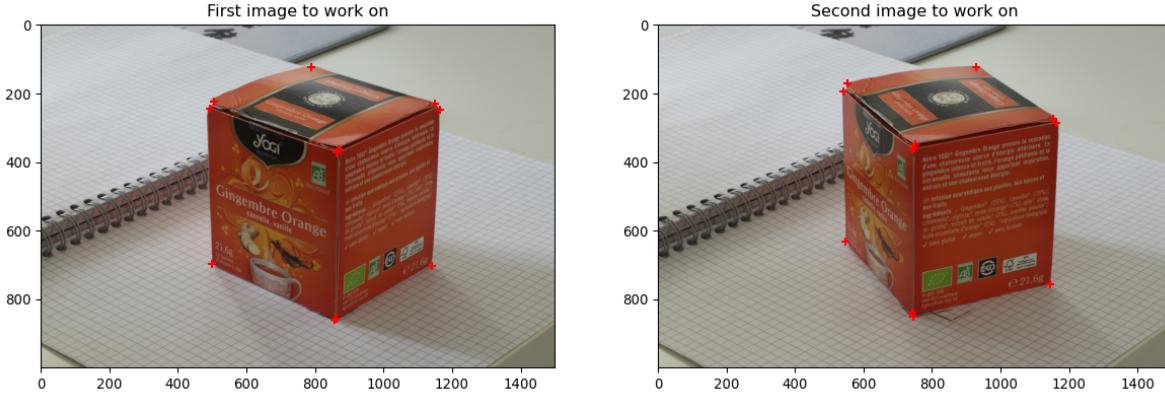


Figure 4: Images de départ pour le calcul de triangulation et de reconstruction.

### 3.3.2 Triangulation d'un point

Grâce aux équations 2 et 3 que l'on peut coupler ensemble on obtient :

$$\begin{cases} \left[ {}^{img}\tilde{p}_g \right] \wedge \left( {}^{img}K_g \ 0 \right) {}^g\tilde{P} = 0 \\ \left[ {}^{img}\tilde{p}_d \right] \wedge \left( {}^{img}K_d \ 0 \right) {}^g\tilde{P} = 0 \end{cases} \quad (4)$$

En supposant que les caméras possèdent les mêmes caractéristiques intrinsèques (ce qui est le cas dans notre étude), on a :

$$\begin{cases} \left[ {}^{img}\tilde{p}_g \right] \wedge \left( {}^{img}K \ 0 \right) {}^g\tilde{P} = 0 \\ \left[ {}^{img}\tilde{p}_d \right] \wedge \left( {}^{img}K \ O \right) {}^dT_g {}^g\tilde{P} = 0 \end{cases} \quad (5)$$

Comme tous les composants de cette équation sont connus (à l'exception de  ${}^g\tilde{P}$  évidemment), on peut réécrire le système d'équation sous une forme matricielle plus simple :

$$\begin{cases} A {}^g\tilde{P} = 0 \\ A' {}^g\tilde{P} = 0 \end{cases} \quad (6)$$

avec  $A$  et  $A'$  deux matrices  $3 \times 4$ . En augmentant le système d'équation, nous pouvons aboutir à une représentation matricielle du système à résoudre :

$$\mathcal{A} {}^g\tilde{P} = 0 \quad (7)$$

**Question 3 :** Comment résoudre cette équation pour aboutir aux coordonnées  ${}^g\tilde{P}$  ?

**Question 4 :** Implémentez une fonction `triangulate(K1, Pt1, K2, Pt2)` prenant en entrée les matrices "caméra" de chacune des vues ainsi que les points en correspondances et retournant le point 3D triangulé.

```
def triangulate(K1, Pt1, K2, Pt2):
    pass

    return pt_t
```

### 3.3.3 Calcul de plan 3D

Une fois la fonction de triangulation implémentée, il ne reste plus qu'à l'utiliser pour trianguler au moins 3 points mis en correspondance dans chaque image et définissant un plan d'une face à reconstruire. Comme la construction d'un plan de l'espace ne nécessite que 2 vecteurs (orthogonaux si possible), choisir 3 points est un minimum. Cependant, nous pouvons affiner le résultat du plan calculé en utilisant plus de points et en effectuant par exemple une optimisation au sens des moindres carrés pour chercher le meilleur plan passant par les points 3D triangulés.

**Question 5 :** Affichez dans une figure les points triangulés ainsi que la pose des caméras utilisées pour le calcul.

Les deux vecteurs directeurs du plan permettent ainsi de remonter directement à l'équation du plan en calculant la normale à celui-ci. Pour ce faire, nous définissons une base en calculant le produit vectoriel entre les deux vecteurs orthogonaux pour obtenir un vecteur normal au plan.

**Question 6 :** En définissant deux vecteur directeurs du plan, calculez et afficher le vecteur normal au plan.

### 3.3.4 Homographie dans le plan 3D

Grâce à la connaissance du plan 3D dans lequel se trouve la face de l'objet à reconstruire, nous pouvons définir une homographie pour coller les points de l'image directement dans celui-ci. Dans un premier temps, et afin de vérifier que l'homographie entre les deux plans est bien effectuée, les points de l'image de base peuvent être collées dans un plan normal de normale  $\vec{z}$  et dont les vecteurs directeurs sont ceux du plan calculé dans la section précédente.

**Question 7 :** Affichez l'homographie d'une face de l'objet dans le plan de normale  $\vec{z}$ .

Une fois cette vérification effectuée, les points de l'image peuvent être reprojetés dans l'espace 3D directement dans le plan 3D de la face. Cela constituera la première face de l'objet reconstruit.

**Question 7 :** Affichez l'homographie d'une face de l'objet dans le plan 3D de la face triangulée.

### 3.3.5 Reconstruction de plusieurs faces

Maintenant qu'il est possible de reconstruire une face d'un objet imagé à partir de plusieurs points mis en correspondance, une étude intéressante consiste à reconstruire plusieurs face d'un même objet imagé. en compilant les commandes effectuées précédemment dans une même fonction, il devient plus aisés d'automatiser la tâche.

L'algorithme de reconstruction de face peut ainsi être décrit comme suit :

1. Extraire 4 points de chaque image en correspondance, définissant un plan.
2. Trianguler chacun des 4 points pour avoir leurs positions dans l'espace.
3. Définir 2 vecteurs définissant un repère dans le plan.
4. En déduire le vecteur normal au plan (définissant ainsi son équation).
5. Effectuer l'homographie des points de l'image vers le plan reconstruit.
6. Calculer la couleur de chaque point du plan (en 3D) grâce à l'homographie et à l'équation du plan.

**Question 8 :** Implémentez une fonction `reconstruct_face(K1, Pts1, K2, Pts2)` prenant en entrée les matrices de caméras utilisées et les points d'une même face mis en correspondance et renvoyant une liste de points 3D

avec une valeur d'intensité (correspondant à l'intensité de chaque pixel considéré lors du calcul).

```
def reconstruct_face(K1, Pts1, K2, Pts2):  
  
    ...  
  
    return points3D, couleurs3D
```

Une fois cette fonction implémentée, il devient plus simple de traiter plusieurs faces simultanément. Cependant l'outil utilisé pour afficher les nuages de points 3D n'est pas optimal. Une solution pour remédier à ce problème est d'utiliser un logiciel plus apte à traiter un grand nombre de points.

**Question 9 :** Affichez la reconstruction d'au moins deux faces de l'objet considéré pendant cette étude.

### 3.4 Pour aller plus loin...

#### 3.4.1 Vers l'estimation de pose

Le travail de reconstruction effectué supposait la connaissance de la pose relative entre les deux caméras. Cependant, lors de la plupart des problèmes de reconstruction, la pose des caméras imageant la scène n'est pas connue. Ainsi la grosse difficulté de la reconstruction en général est de connaître les paramètres extrinsèques des caméras.

*Question 10 :* Décrivez le processus de détermination des paramètres extrinsèques d'un système de caméras.

A partir de ce processus il devient dès lors possible de reconstruire plus aisément des objets imagés à partir de plusieurs caméras.

#### 3.4.2 Un autre moyen de reconstruire un objet 3D ?

Cependant, bien les paramètres extrinsèques des caméras soient connus, effectuer la triangulation entre chaque point connu de la scène devient rapidement fastidieux et lourd à implémenter. Pour remédier à cela, la géométrie épipolaire offre des alternatives plus calculatoires, mais qui permettent d'aboutir rapidement à la reconstruction de points en 3D à partir de deux points imagés dans deux caméras distinctes. Les applications utilisant ces méthodes de reconstructions sont nombreuses et constituent un vrai défi aujourd'hui en terme de recherche et ce notamment dans un domaine très en vogue ces temps-ci.

*Question 11:* De quelle méthode s'agit-il et sur quoi se base-t-elle pour fonctionner ?

La reconstruction 3D d'objets n'a pas que des applications purement de recherche ou industrielles et un domaine en essor tire son épingle du jeu en offrant des textures de plus en plus réalistes (car tirées de scènes réelles).

*Question 12:* De quel domaine s'agit-il ? Citez un exemple utilisant directement la reconstruction 3D comme base graphique.



Figure 5: Résultat d'une reconstruction par photogrammétrie dans une scène 3D CAO.