Calibrage de 2 caméras à partir de points de correspondances

Calibrer deux caméras à partir d'un ensemble de points de correspondance est un problème classique de vision par ordinateur traité ici avec une approche non-linéaire. La méthode proposée est une extension des méthodes de rectification épipôlaires d'images consistant à faire en sorte que chaque droite épipôlaire soit orientée dans le sens des lignes de l'images.

1 Position du problème

Etant données deux images de la même scène, prises de deux points de vues distincts ainsi qu'un ensemble de points de correspondance entre ces deux images, le problème traité dans ce projet consiste à calibrer les deux caméras, c'est-à-dire à définir les paramètres intrinsèques et extrinsèques des deux caméras. Une fois les caméras calibrées, il est alors possible de faire une reconstruction 3D des points de correspondance par triangulation.

Deux images de la même scène et la reconstruction en vue de dessus.
Les parties en rouges représentent les deux caméras.

Il existe plusieurs techniques pour calibrer deux caméras à partir de points de correspondances. La grande majorité des méthodes existantes sont linéaires, rapides, mais relativement instables. La technique proposée dans ce sujet est une méthode itérative où le système converge vers la solution de calibrage des caméras, ce qui en fait une méthode lente, mais plutôt robuste.

2 Modélisation d'une caméra

En vision par ordinateur, une caméra est modélisée par une matrice de projection $\mathtt{P}_{3\times4}$ telle qu'un point 3d $\mathbf{X}$ se projette sur le pixel $\mathbf{x} = \mathtt{P}\mathbf{X}$. Une matrice de projection se décompose en trois éléments : la matrice $\mathtt{K}$ des paramètres intrinsèques, la matrice de rotation $\mathtt{R}$ spécifiant l'orientation de la caméra et la position $\mathbf{c}$ du centre de projection de la caméra.

À noter que les points finis $\mathbf{X}$ et $\mathbf{x}$ sont exprimés en coordonnées homogènes.

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ w \end{pmatrix} \doteq \begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix} \doteq \begin{pmatrix} kx \\ ky \\ kw \end{pmatrix} \quad \forall k \neq 0.$$

$$\mathbf{X} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \doteq \begin{pmatrix} x/w \\ y/w \\ z/w \\ 1 \end{pmatrix} \doteq \begin{pmatrix} kx \\ ky \\ kz \\ kw \end{pmatrix} \quad \forall k \neq 0.$$

2.1 Paramètres intrinsèques

La matrice des paramètres intrinsèques se compose de la façon suivante :

$$\mathtt{K} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- f est la focale de la caméra exprimée en pixels. Lorsque la focale d'une caméra est inconnue, il est courant d'en faire une approximation par la valeur $f = \sqrt{w^2 + h^2}$ où w et h représentent respectivement la largeur et la hauteur de l'image, exprimées en pixels.
- x_0 et y_0, exprimés en pixels, représentent les coordonnées du point principal correspondant à la position de l'intersection de l'axe optique et du plan image. Il est d'usage d'assimiler ce point au centre de l'image : $x_0 = w/2$ et $y_0 = h/2$.

2.2 Paramètres extrinsèques

La position et l'orientation de la caméra sont exprimées par une matrice de rotation $\mathtt{R}_{3\times3}$ et par un vecteur position $\mathbf{c}$. Celui-ci correspond à la position du centre de projection de la caméra dans le repère de la scène.

$$\mathtt{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{et} \quad \mathbf{c} = \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix}$$

2.3 La caméra

La matrice de projection de la caméra est une combinaison des trois paramètres décrits précédemment :

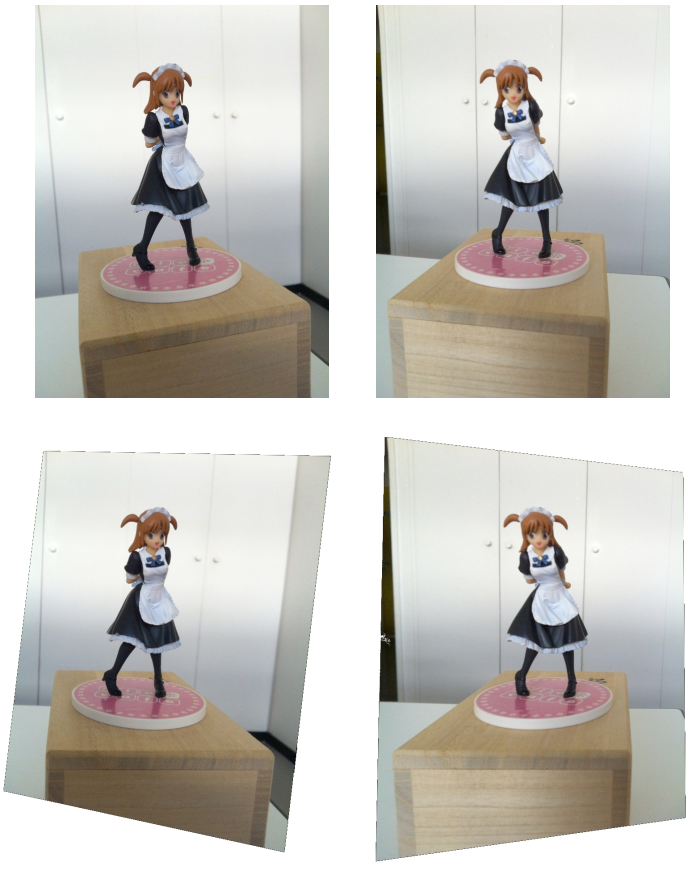
$$\mathtt{P} = \mathtt{K}\left[\mathtt{R}\middle|-\mathtt{R}\mathbf{c}\right]$$

$\mathtt{P}$ est donc une matrice 3×4 permettant de projeter un point 3d $\mathbf{X}$ sur le pixel $\mathbf{x} = \mathtt{P}\mathbf{X}$. Si l'on connait les matrices $\mathtt{K}$ et $\mathtt{R}$ ainsi que le vecteur $\mathbf{c}$, il est alors possible de constuire la matrice de projection $\mathtt{P}$ de la caméra.

3 Calibrage des caméras

3.1 Introduction

L'approche présentée dans ce document est une extension des méthodes de rectification épipôlaires consistant à transformer les deux images par des homographies de telle sorte que chaque point d'une image ait son correspondant sur l'autre image alignée sur la même ligne.



Rectification épipôlaire.

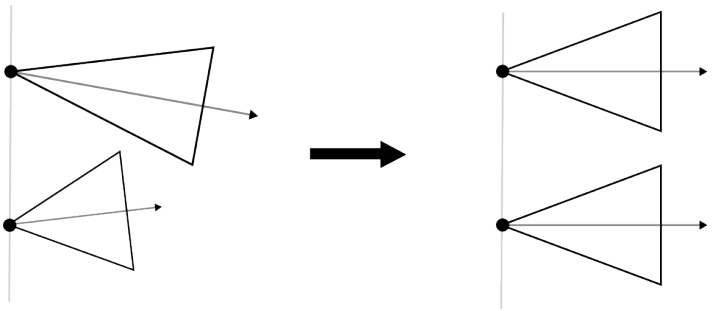
Il s'agit donc de trouver une transformation des caméras telle qu'elles prennent une forme connue avec :

- les axes optiques des caméras parallèles
- la même focale pour les deux caméras
- l'axe des x des caméras orientés selon la droite passant par les centres des caméras

Les matrices de projection $\mathtt{P}_1^{\blacktriangledown}$ et $\mathtt{P}_2^{\blacktriangledown}$ de ces caméras transformées peuvent s'écrire :

$$\mathtt{P}_1^{\blacktriangledown} = \mathtt{K}\,[\mathtt{Id}|-\mathbf{c}_1] \quad \text{et} \quad \mathtt{P}_2^{\blacktriangledown} = \mathtt{K}\,[\mathtt{Id}|-\mathbf{c}_2]$$

en prenant arbitrairement $\mathbf{c}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ et $\mathbf{c}_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$



transformation des caméras.

Connaissant d'une part la forme finale des caméras et d'autre part la transformation appliquée sur les caméras initiales pour qu'elles prennent cette forme finale, il est possible de trouver les paramètres des caméras initiales en appliquant aux caméras finales la transformation inverse.

3.2 Transformation des caméras

Le but des transformations sur les deux images initiales est de faire en sorte que chaque point d'une image soit aligné avec son correspondant sur l'axe des x de l'image. Les transformations appliquées sur les images sont calculées à partir d'opérations réalisables sur des caméras, à savoir un changement de focale et un changement d'orientation. Dans notre cas, nous estimons que les deux caméras ont la même focale. Par conséquent, la transformation d'une caméra ne concernera que son orientation. Pour modéliser une rotation $\mathtt{R}$ de la caméra, nous appliquons sur l'image l'homographie $\mathtt{H}$ définie de la manière suivante :

$$\mathtt{H} = \mathtt{K}\mathtt{R}\mathtt{K}^{-1}$$

Cette homographie représente l'annulation de la projection $\mathtt{K}^{-1}$ suivie de la rotation souhaitée puis de la projection $\mathtt{K}$. Reste encore à trouver la bonne paire de rotations $\mathtt{R}_1$ et $\mathtt{R}_2$ permettant de rectifier chacune des deux images. C'est justement cette rotation que nous allons chercher

"à tâtons" avec une approche non-linéaire. Une rotation $\mathtt{R}$ peut s'exprimer sous la forme d'Euler qui consite à décomposer cette rotation selon les trois axes :

$$\mathtt{R}_E(\theta_x, \theta_y, \theta_z) = \mathtt{R}_x(\theta_x)\mathtt{R}_z(\theta_z)\mathtt{R}_y(\theta_y)$$

avec :

$$\mathtt{R}_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin(\theta_x) & cos(\theta_x) \end{bmatrix}$$

$$\mathtt{R}_y(\theta_y) = \begin{bmatrix} cos(\theta_y) & 0 & -sin(\theta_y) \\ 0 & 1 & 0 \\ sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix}$$

$$\mathtt{R}_z(\theta_z) = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Donc finalement, on cherche les deux matrices de rotations $\mathtt{R}_1 = \mathtt{R}_E(\theta_x^1, \theta_y^1, \theta_z^1)$ et $\mathtt{R}_2 = \mathtt{R}_E(\theta_x^2, \theta_y^2, \theta_z^2)$ permettant de rectifier les deux images. Pour stabiliser le processus de rectification des caméras, il est judicieux de bloquer la rotation autour de l'axe des x pour l'une des deux caméras, en fixant par exemple $\theta_x^1 = 0$.

La transformation permettant de rectifier les deux images est finalement donnée par les deux matrices d'homographies $\mathtt{H}_1$ et $\mathtt{H}_2$:

$$\mathtt{H}_1 = \mathtt{K}\mathtt{R}_1\mathtt{K}^{-1} \quad \text{et} \quad \mathtt{H}_2 = \mathtt{K}\mathtt{R}_2\mathtt{K}^{-1}$$

avec

$$\mathtt{R}_1 = \mathtt{R}_E(0, \theta_y^1, \theta_z^1) \quad \text{et} \quad \mathtt{R}_2 = \mathtt{R}_E(\theta_x^2, \theta_y^2, \theta_z^2)$$

3.3 Minimisation

Le but du processus de rectification d'image est de minimiser la distance sur l'axe des y entre un point sur une image et son correspondant sur l'autre image. En d'autres termes, on cherche à minimiser un résidu R^2 correspondant à la somme des disparités verticales pour chaque point de correspondance. Soit l'ensemble de points de correspondances $\{\mathbf{x}_i^1 \leftrightarrow \mathbf{x}_i^2\}$ entre l'image 1 et l'image 2 et soient $\mathtt{H}_1$ et $\mathtt{H}_2$ les deux matrices de transformation candidates pour rectifier les images, le résidu R^2 est de la forme suivante :

$$R^2 = \sum_i \left((\mathtt{H}_1\mathbf{x}_i^1)_y - (\mathtt{H}_2\mathbf{x}_i^2)_y \right)^2$$

où $(\mathbf{a})_y$ correspond à la composante y du vecteur $\mathbf{a}$, en veillant à ce que la composante homogène du vecteur $\mathbf{a}$ soit égale à 1. Un des objectif de notre programme sera donc de trouver les matrices $\mathtt{H}_1$ et $\mathtt{H}_2$ qui minimisent le résidu R^2. Ce problème revient en fin de compte à trouver les angles $\theta_y^1, \theta_z^1, \theta_x^2, \theta_y^2$ et θ_z^2 permettant de calculer ces deux homographies.

3.4 Calibrage des caméras

Une fois trouvées les deux matrices de rotations $\mathtt{R}_1$ et $\mathtt{R}_2$ permettant de rectifier les deux images, reste encore à calibrer les caméras. On sait que les caméras rectifiées ont la forme suivante :

$$\mathtt{P}_1^{\blacktriangledown} = \mathtt{K}\,[\mathtt{Id}|-\mathbf{c}_1] \quad \text{et} \quad \mathtt{P}_2^{\blacktriangledown} = \mathtt{K}\,[\mathtt{Id}|-\mathbf{c}_2]$$

et que pour arriver à cette forme, elles ont subi respectivement les rotations $\mathtt{R}_1$ et $\mathtt{R}_2$. Pour calibrer les caméras, il suffit donc d'appliquer les rotations inverses aux matrices finales :

$$\mathtt{P}_1 = \mathtt{K}\left[\mathtt{R}_1^\top|-\mathtt{R}_1^\top\mathbf{c}_1\right] \quad \text{et} \quad \mathtt{P}_2 = \mathtt{K}\left[\mathtt{R}_2^\top|-\mathtt{R}_2^\top\mathbf{c}_2\right]$$

4 Systèmes non-linéaires

4.1 Les paramètres

Un phénomène est dit non-linéaire lorsque des grandeurs caractéristiques du phénomène reliées entre elles ne varient pas proportionnellement l'une à l'autre. Il existe plusieurs sortes de problèmes non-linéaires et plusieurs façons de les traiter. La plupart des méthodes partent d'une solution initiale que l'on espère pas trop éloignée de la solution recherchée, puis affinent itérativement cette solution de départ pour se raprocher de la solution optimale. Le problème non-linéaire que nous allons traiter peut s'écrire sous la forme $R^2 = |f(\mathbf{a},\mathbf{b})|$ où :

- f est la fonction que l'on souhaite traiter.
- On veut minimiser $R^2 = |f(\mathbf{a},\mathbf{b})|$, autrement dit, on veut faire tendre f vers 0 autant que possible.
- $\mathbf{a}$ est un vecteur contenant toutes les variables que nous pouvons modifier pour faire tendre f vers 0.
- $\mathbf{b}$ est un vecteur contenant toutes les constantes représentant le système modélisé. Ainsi, l'évaluation de f nécessite les variables $\mathbf{a}$ et les constantes du système $\mathbf{b}$.

Dans le cas du calibrage des caméras, nous considérons que les deux caméras ont la même focale. Le vecteur $\mathbf{a}$ correspond alors aux angles de rotation des caméras :

$$\mathbf{a} = \begin{pmatrix} \theta_y^1 \\ \theta_z^1 \\ \theta_x^2 \\ \theta_y^2 \\ \theta_z^2 \end{pmatrix}$$

Le vecteur $\mathbf{b}$ contient toutes les informations constantes du système, c'est à dire la focale f des caméras, le point principal des caméras (x_0, y_0), le nombre de points de correspondance

ainsi que la liste de ces points de correspondance.

$$\mathbf{b} = \begin{pmatrix} f \\ w/2 \\ h/2 \\ \texttt{nbPoints} \\ x_1 \\ y_1 \\ x_2 \\ y_2 \\ x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \end{pmatrix}$$

En pratique, la fonction $f(\mathbf{a}, \mathbf{b})$ de votre programme extrait les paramètres du vecteur $\mathbf{b}$ et utilise les variables du vecteur $\mathbf{a}$ pour calculer le résidu R^2.

4.2 La matrice jacobienne

La matrice jacobienne est la matrice des dérivées partielles du premier ordre d'une fonction vectorielle. Dans le système non-linéaire $f(\mathbf{a}, \mathbf{b})$, c'est la matrice $\mathtt{J}$ dont chaque élément est défini par :

$$\mathtt{J}_{ij} = \frac{\partial f_i}{\partial a_j}$$

Dans notre cas, la fonction f renvoie un scalaire, la matrice jacobienne est donc une matrice à une seule ligne dont le j^{eme} élément s'écrit :

$$\mathtt{J}_j = \frac{\partial f}{\partial a_j}$$

La matrice jacobienne permet d'estimer la variation du résidu R^2 du système si l'on fait varier légèrement l'un des coefficients θ_y^1, θ_z^1, ... θ_z^2.

En pratique, une dérivée partielle se calcule soit à partir de l'expression formelle de la dérivée de la fonction traitée si elle est connue, soit en l'estimant numériquement en faisant varier de δa_j le j^{eme} élément de $\mathbf{a}$:

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_j \\ \vdots \\ a_n \end{pmatrix} \qquad \mathbf{a_j} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_j + \delta a_j \\ \vdots \\ a_n \end{pmatrix}$$

L'élément $\mathtt{J}_j$ se calcule alors comme suit :

$$\mathtt{J}_j = \frac{\partial f}{\partial a_j} \simeq \frac{f(\mathbf{a_j}, \mathbf{b}) - f(\mathbf{a}, \mathbf{b})}{\delta a_j}$$

Dans la littérature, il est conseillé de choisir $\delta a_j = \mathtt{max}\left(\mathtt{min}(|a_j|.10^{-4}, 10^{-6}), 10^{-15}\right)$. Ceci permet d'essayer de prendre une valeur de $\delta a_j = a_j.10^{-4}$ tout en imposant que δa_j soit compris entre 10^{-6} et 10^{-15}.

4.3 Le pas $\mathbf{\Delta a}$

Le but de l'étape suivante est d'utiliser la matrice jacobienne calculée précédement afin de trouver la meilleur variation $\mathbf{\Delta a}$ de $\mathbf{a}$ permettant de minimiser $R^2 = |f(\mathbf{a}, \mathbf{b})|$. Les méthodes de résolution de systèmes non linéaires sont généralement basées sur l'hypothèse que la fonction f a localement un comportement linéaire. Elle peut donc être approximée en $\mathbf{a}$ par :

$$f(\mathbf{a} + \mathbf{\Delta a}, \mathbf{b}) \simeq f(\mathbf{a}, \mathbf{b}) + \mathtt{J}\mathbf{\Delta a}$$

Dans notre cas, nous cherchons un $\mathbf{\Delta a}$ tel que $f(\mathbf{a} + \mathbf{\Delta a}, \mathbf{b}) = 0$. L'équation précédente nous donne alors :

$$f(\mathbf{a}, \mathbf{b}) + \mathtt{J}\mathbf{\Delta a} = 0$$

soit

$$\mathtt{J}\mathbf{\Delta a} = -f(\mathbf{a}, \mathbf{b})$$

Ce qui peut se résoudre par :

$$\mathbf{\Delta a} = -\mathtt{J}^{+} f(\mathbf{a}, \mathbf{b})$$

où $\mathtt{J}^{+} = (\mathtt{J}^{\top}\mathtt{J})^{-1}\mathtt{J}^{\top}$ est la matrice pseudo-inverse de $\mathtt{J}$.

Un pas $\mathbf{\Delta a}$ est acceptable s'il satisfait effectivement $|f(\mathbf{a} + \mathbf{\Delta a}, \mathbf{b})| < |f(\mathbf{a}, \mathbf{b})|$, ce qui n'est pas nécessairement le cas selon f et $\mathbf{a}$. Une bonne valeur de $\mathbf{\Delta a}$ doit d'une part ne pas être trop petite, sans quoi la résolution du système est trop longue et le risque de tomber dans un minimum local augmente. D'autre part, cette valeur ne doit pas non plus être trop grande car le risque serait alors de dépasser le minimum global et de diverger. La solution à ce problème est de pondérer le pas $\mathbf{\Delta a}$ par une valeur λ tel qu'il soit ni trop petit, ni trop grand. On choisit d'abord une valeur initiale de λ. Dans la littérature, il est conseillé de choisir λ comme 10^{-3} fois la valeur moyenne des éléments de la diagonale de $N = \mathtt{J}^{\top}\mathtt{J}$. Le calcul de $\mathbf{\Delta a}$ se fait alors en résolvant à l'aide de la SVD le système suivant :

$$(\mathtt{J}^{\top}\mathtt{J} + \lambda Id)\mathbf{\Delta a} = -\mathtt{J}^{\top} f(\mathbf{a}, \mathbf{b})$$

Si le $\mathbf{\Delta a}$ obtenu fait converger f, alors on met à jour $\mathbf{a}$ et on divise λ par 10 pour que le pas de la prochaine itération soit plus grand. Si le $\mathbf{\Delta a}$ obtenu ne fait pas converger f, alors le pas était trop grand et on ré-écrit le système précédent avec une valeur de λ multipliée par 10. On répète ce processus jusqu'à ce qu'on obtienne un λ acceptable.

4.4 En pratique

La stabilité numérique des méthodes de résolution de systèmes non linéaires dépend très souvent de la qualité des conditions initiales. Dans notre cas, les deux images devront être prises avec le même appareil et devront avoir la même résolution $w \times h$. Le processus de minimisation non-linéaire peut alors partir des conditions initiales suivantes :

- le point principal (x_0, y_0) correspond au centre des images et la focale f des caméras est estimée avec la formule $f = \sqrt{w^2 + h^2}$
- au départ, les coefficients $\theta_y^1, \theta_z^1, \theta_x^2, \theta_y^2$ et θ_z^2 sont tous nuls.

Le processus itératif de résolution du système linéaire est décrit en détail dans l'algorithme 1.

Algorithme 1: Résolution de systèmes non linéaires

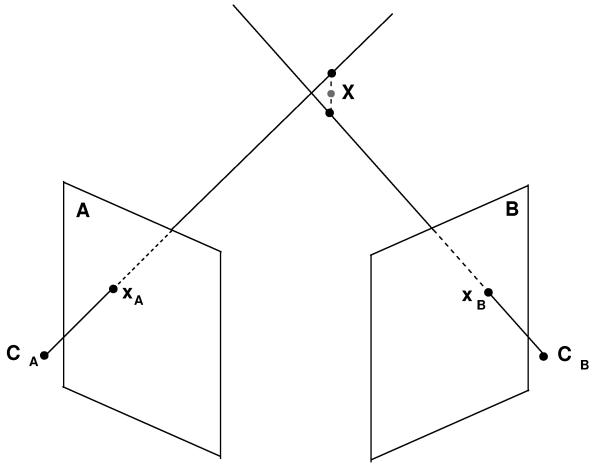
Data : Un vecteur $\mathbf{a}$ des variables à calculer, un vecteur $\mathbf{b}$ des constantes du système et une fonction $f(\mathbf{a}, \mathbf{b})$ à minimiser.

Result : une valeur de $\mathbf{a}$ telle que f soit minimal.

$\mathtt{J} = [\partial f / \partial \mathbf{a}]$
$\lambda = \text{average}\{\text{diag}(\mathtt{J}^\top \mathtt{J})\}.10^{-3}$
foreach *iteration* **do**
  $\mathtt{J} = [\partial f / \partial \mathbf{a}]$
  compteur $= 0$
  accepté $=$ false
  repeat
    résoudre : $(\mathtt{J}^\top \mathtt{J} + \lambda Id)\mathbf{\Delta a} = -\mathtt{J}^\top f(\mathbf{a}, \mathbf{b})$
    if $|f(\mathbf{a} + \mathbf{\Delta a}, \mathbf{b})| < |f(\mathbf{a}, \mathbf{b})|$ **then**
      $\mathbf{a} = \mathbf{a} + \mathbf{\Delta a}$
      $\lambda = \lambda / 10$
      accepté $=$ true
    end
    else
      $\lambda = \lambda \times 10$
    end
    compteur $=$ compteur $+ 1$
    if *compteur* > 100 **then**
      return a
    end
  until *accepté* $=$ true;
end
return a

5 Triangulation 3d

Etant donnée une paire de points de correspondance $\mathbf{x}_1$ et $\mathbf{x}_2$ et les matrices de projection $\mathtt{P}_1$ et $\mathtt{P}_2$ des caméras, la triangulation de $\mathbf{x}_1$ et $\mathbf{x}_2$ consiste à calculer l'intersection des rayons issus de chaque caméra passant par ces deux points respectivement. Evidement, il est peu probable que les deux rayons s'intersectent parfaitement, on cherchera donc le point d'intersection au sens des moindres carrés.



Triangulation 3d.

D'un point de vue plus formel, on sait que le point $\mathbf{X}$ recherché se projette sur le pixel $\mathbf{x}_1$ de l'image 1 et sur le pixel $\mathbf{x}_2$ de l'image 2, autrement dit :

$$\mathbf{x}_1 = \mathtt{P}_1\mathbf{X} \quad \text{et} \quad \mathbf{x}_2 = \mathtt{P}_2\mathbf{X}$$

Nous obtenons alors :

$$\mathbf{x}_1 \times \mathtt{P}_1\mathbf{X} = \mathbf{0} \quad \text{et} \quad \mathbf{x}_2 \times \mathtt{P}_2\mathbf{X} = \mathbf{0}$$

(où $\times$ est le produit vectoriel de $\mathbb{R}^3$).

En ré-écrivant les matrices de projection par leurs vecteurs lignes :

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \quad \leftrightarrow \quad P = \begin{bmatrix} \mathbf{p}_{1\bullet}^\top \\ \mathbf{p}_{2\bullet}^\top \\ \mathbf{p}_{3\bullet}^\top \end{bmatrix}$$

où $\mathbf{p}_{i\bullet}^\top$ correspond à la i^{eme} ligne de $\mathtt{P}$.

Chacune des deux équations précédentes prend la forme :

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \times \begin{pmatrix} \mathbf{p}_{1\bullet}^\top\mathbf{X} \\ \mathbf{p}_{2\bullet}^\top\mathbf{X} \\ \mathbf{p}_{3\bullet}^\top\mathbf{X} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

soit

$$\begin{pmatrix} y\mathbf{p}_{3\bullet}^{\top}\mathbf{X} - w\mathbf{p}_{2\bullet}^{\top}\mathbf{X} \\ w\mathbf{p}_{1\bullet}^{\top}\mathbf{X} - x\mathbf{p}_{3\bullet}^{\top}\mathbf{X} \\ x\mathbf{p}_{2\bullet}^{\top}\mathbf{X} - y\mathbf{p}_{1\bullet}^{\top}\mathbf{X} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

qui peut s'écrire sous forme matricielle :

$$\begin{bmatrix} y\mathbf{p}_{3\bullet}^{\top} - w\mathbf{p}_{2\bullet}^{\top} \\ w\mathbf{p}_{1\bullet}^{\top} - x\mathbf{p}_{3\bullet}^{\top} \\ x\mathbf{p}_{2\bullet}^{\top} - y\mathbf{p}_{1\bullet}^{\top} \end{bmatrix} \mathbf{X} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

En considérant que la matrice décrite ici est de rang 2 et que cette équation est valable pour les deux caméras, on peut finalement écrire l'équation :

$$\begin{bmatrix} y_1\mathbf{p}_{(1),3\bullet}^{\top} - w_1\mathbf{p}_{(1),2\bullet}^{\top} \\ w_1\mathbf{p}_{(1),1\bullet}^{\top} - x_1\mathbf{p}_{(1),3\bullet}^{\top} \\ y_2\mathbf{p}_{(2),3\bullet}^{\top} - w_2\mathbf{p}_{(2),2\bullet}^{\top} \\ w_2\mathbf{p}_{(2),1\bullet}^{\top} - x_2\mathbf{p}_{(2),3\bullet}^{\top} \end{bmatrix} \mathbf{X} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

où $\mathbf{p}_{(c),i\bullet}$ correspond à la i^{eme} ligne de la matrice de projetction de la c^{eme} caméra. Ainsi, les deux premières lignes concernent la matrice $\mathtt{P}_1$ et le point $\mathbf{x}_1$ alors que les deux dernières lignes concernent la matrice $\mathtt{P}_2$ et le point $\mathbf{x}_2$. Il suffit de résoudre ce système au sens des moindres carrés (avec une SVD) pour trouver le point triangulé $\mathbf{X}$.

6 Travail demandé

Vous devez réaliser par groupe de deux étudiants un programme en langage `C++` à l'aide de la bibliothèque **OpenKraken**. Vous séparerez vos programmes dans des fichiers `.cpp` et `.hpp` qui seront compilés à l'aide d'un makefile. Votre programme devra compiler sans warning (mais avec l'option `-Wall`).

Vous rendrez votre projet sous forme d'archive nommée `nom1_nom2_calibration.tgz`. Cette archive doit générer un repertoire nommé `nom1_nom2_calibration/` contenant votre projet. Le répertoire en question doit contenir votre code ainsi qu'une version électronique de votre rapport.

6.1 Validation de votre projet

Votre programe devra pouvoir charger deux images et deux liste de points. Votre programme devra ensuite calibrer les deux caméras et faire une reconstruction 3D des points triangulés.

Afin de valider facilement votre programme, vous créerez un répertoire `output/` dans lequel vous stockerez une image `reconstruction.jpg` représantant une reconstruction 3D de la scène vue de dessus.

6.2 Options de commandes et autres contraintes

Il est important de respecter les notations spécifiées dans ce sujet, notamment le nom des options détaillées ci-dessous. Pour commencer, votre éxécutable s'appellera : `./calibration`. Ensuite, les options suivantes sont obligatoires :

- `-h` pour afficher l'aide
- `-i [image]*.jpge` pour spécifier les noms des images à traiter.
- `-l [image]*.list` pour spécifier les noms des listes contenant les points.

Vous pouvez tout à fait rajouter d'autres options dont vous détaillerez la fonction dans l'aide.

Attention, votre programme ne doit pas contenir de `std::cin` ou de `scanf`.

Enfin, votre fonction de résolution de systèmes non linéraires prendra en paramètre la fonction à minimiser sous forme de pointeur de fonction. Son prototype sera :

```
void nonLinearSystemSolver(
        kn::Vector<double> &a,
        const kn::Vector<double> &b,
        double (*costFunctionPtr)(const kn::Vector<double>&,const kn::Vector<double>&),
        const unsigned int nbMaxIteration);
```

A noter que pour des questions de généricité, toutes les données constantes quel que soit leur type doivent être stockées sous la forme d'un vecteur `b` de `double`.

6.3 Formats de fichier

Afin de faciliter l'édition des points de correspondance, vous pourrez utiliser un programme indépendant de sélection de points. Chaque liste de points sera ainsi stocké dans un fichier `.list`.

6.4 Rapport

Vous fournirez en version papier et en version électronique un rapport de 10 pages maximum. **Consacrez suffisament de temps au rapport car il représente une bonne partie de la note finale**. Essayez de respecter au mieux les directives suivantes :

- **Votre rapport doit comprendre une page résumant vos travaux avec les indications suivantes** :
  - éléments demandés et codés qui fonctionnent.
  - éléments demandés et codés qui ne fonctionnent pas.
  - éléments demandés mais pas codés.
  - éléments non demandés et codés qui fonctionnent.
  - éléments non demandés mais pas codés ou qui ne fonctionnent pas.
- Ne perdez pas de temps à réexpliquer le sujet du projet, l'enseignant le connaît déjà, faites seulement un bref résumé de quelques lignes. De manière plus générale, ne détaillez pas des méthodes déjà expliquées dans l'énoncé à moins que vous les ayez modifiées.

– Un rapport sert surtout à montrer comment vous avez fait face aux problèmes (d'ordre algorithmique). Certains problèmes sont connus (on en parle dans l'énoncé), d'autres sont imprévus. Montrez que vous les avez remarqués et compris. Donnez la liste des solutions à ce problème et indiquez votre choix. Justifiez votre choix (vous avez le droit de dire que c'est la méthode la plus facile à coder).
– Il ne doit figurer aucune ligne de code dans votre rapport. Un rapport n'est pas un listing de votre programme où vous détaillez chaque fonction. Vous devez par contre détailler vos structures de données et mettre du pseudocode pour expliquer vos choix algorithmiques.
  Il est autorisé d'utiliser des "raccourcis" tels que "`initialiser le tableau tab à 0`" plutôt que de détailler la boucle faisant la même chose.
– n'hésitez pas à mettre des images dans votre rapport pour illustrer vos propos et vos résultats.
– Enfin, il est **très important** de faire la liste de ce que vous avez fait, de ce qui fonctionne correctement et la liste des dysfonctionnements. Précisez quand il s'agit d'options que vous avez rajoutées en plus de ce qui était demandé.

7 Pour finir

Vous pouvez laisser libre cours à votre imagination, toute amélioration sera la bienvenue. Vous trouverez quelques informations complémentaires à l'adresse :

`http://www-igm.univ-mlv.fr/~vnozick/`

Bon courage.