# Projet de mathématiques

**Mathématiques pour l'informatique**

*—IMAC 2—*

---

## Lignes de fuite

Ce projet est une introduction aux problèmes de vision par ordinateur et de géométrie projective. Il inclut d'une part l'utilisation d'outils mathématiques tels que la résolution de systèmes linéaires surdéterminés au sens des moindres carrés, et d'autre part de la programation en C++ et en OpenGL.

Ce projet est à réaliser par groupe de 2 en C++.

---

## 1 Introduction

Extraire de l'information 3d à partir d'une photographie peut paraître trivial à n'importe quelle personne alors qu'un ordinateur peine à en extraire le moindre indice de géométrie 3d. En effet, alors même qu'un enfant identifie des objets et reconstruit mentalement la scène photographiée, l'ordinateur n'y "voit" que des pixels.

Les outils de vision par ordinateur sont particulièrement performants quand sont appliqués à plusieurs images représentant la même scène. Il existe toutefois des méthodes permettant d'extraire de l'information 3d à partir d'une seule image. Il s'agit d'indices monoculaires comme les ombres, les occlusions, l'endroit où l'on fait le focus, la texture des objets, etc. Cependant le plus évident reste l'étude des lignes de fuite et des points de fuite (*vanishing lines* et *vanishing points* en anglais).
Leur étude permet d'établir des caractèristiques géométriques de la scène comme par exemple la taille relative de deux objets orientés vers le même point de fuite. L'étude des lignes de fuite permet aussi de déterminer les paramètres optiques de la caméra. Avec un petit effort supplémentaire, il est aussi possible de déterminer les paramètres de position et d'orientation de la caméra. Dans ce projet, nous appliquerons le calibrage de la caméra à la réalité augmentée : ajouter de l'image de synthèse sur l'image d'une scène réelle.

De façons plus exhaustive, ce sujet aborde les points suivants :
- Calcul des points de fuite et des lignes de fuite.
- Calcul de l'image de la conique absolue.
- Calibrage de la caméra et réalité augmentée.

# 2 Notions mathématiques

Ce chapitre regroupe quelques notions de géométrie projective et d'algèbre linéaire nécessaires à la mise en oeuvre de ce projet.

## 2.1 Géométrie projective

La géométrie projective rajoute une coordonnée de plus par rapport à la géométrie euclidienne. Alors qu'un point dans $\mathbb{R}^2$ n'a que 2 dimensions, ce même point en aura 3 dans $\mathbb{P}^2$ (coordonnées homogènes). Dans $\mathbb{P}^2$, un point se note $\mathbf{x} = (x, y, w)^\top$.

Pour passer des coordonnées euclidiennes $\mathbf{x}_{\mathbb{R}^2} = (x', y')^\top$ aux coordonnées homogènes $\mathbf{x}_{\mathbb{P}^2} = (x, y, w)^\top$, il suffit d'ajouter une composante $w = 1$ à $\mathbf{x}_{\mathbb{R}^2}$ qui devient $\mathbf{x}_{\mathbb{P}^2} = (x', y', 1)^\top$. Pour passer des coordonnées homogènes $\mathbf{x}_{\mathbb{P}^2} = (x, y, w)^\top$ aux coordonnées euclidiennes $\mathbf{x}_{\mathbb{R}^2} = (x', y')^\top$, il suffit de diviser chaque composantes de $\mathbf{x}_{\mathbb{P}^2}$ par $w$ (si $w \neq 0$). On obtient alors $\mathbf{x}_{\mathbb{R}^2} = (x', y')^\top = (\frac{x}{w}, \frac{y}{w})^\top$. Si $w = 0$, alors $x' = x$ et $y' = y$ mais $\mathbf{x}_{\mathbb{R}^2}$ représente alors un point à l'infini, qui correspond à une direction plutôt qu'à un point.

Voici quelques propriétés intéressantes des points dans $\mathbb{P}^2$ :

- Les points $\mathbf{x}$ et $k\mathbf{x}$ ($k \in \mathbb{R}^*$) représentent le même point.

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \doteq \begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix} \doteq \begin{pmatrix} kx \\ ky \\ kw \end{pmatrix}$$

  où $x/w$ et $y/w$ représentent les coordonnées cartésiennes du point pour $w \neq 0$ et '$\doteq$' correspond à une égalité à un facteur d'échelle près.
- Une matrice transformant un point de $\mathbb{P}^2$ est elle aussi invariante par facteur d'échelle.
- Une droite de $\mathbb{P}^2$ se note $\mathbf{l} = (a, b, c)^\top$.
- Les droites $\mathbf{l}$ et $k\mathbf{l}$ ($k \neq 0$) représentent la même droite.
- Un point $\mathbf{x} = (x, y, w)^\top \in \mathbf{l}$ ssi $\mathbf{x}^\top \mathbf{l} = \mathbf{l}^\top \mathbf{x} = 0$.
- La droite $\mathbf{l}$ passant par les points $\mathbf{x}_1$ et $\mathbf{x}_2$ vaut : $\mathbf{l} \doteq \mathbf{x}_1 \times \mathbf{x}_2$ (où $\times$ correspond au produit vectoriel de $\mathbb{R}^3$).
- L'intersection $\mathbf{x}$ de 2 droites $\mathbf{l}_1$ et $\mathbf{l}_2$ se calcule avec : $\mathbf{x} \doteq \mathbf{l}_1 \times \mathbf{l}_2$. Notez le caractère dual entre les points et les droites dans $\mathbb{P}^2$.
- Le point $(x, y, 0)^\top$ correspond à un point à l'infini aussi appelé point idéal.
- Deux droites parallèles s'intersectent en un point à l'infini.

## 2.2 Systèmes surdéterminés

Un système surdéterminé est un système où il y a plus d'équations que d'inconnues. Les équations supplémentaires ne sont pas nécessairement en accord avec les précédentes (elles ne sont pas forcément combinaisons linéaires des premières). Les solutions trouvées ne satisferont donc pas nécessairement toutes les équations mais tenteront d'en satisfaire un maximum (cf. méthode des moindres carrés).

Pour résoudre un système surdéterminé $A\mathbf{x} = \mathbf{b}$, on peut utiliser la matrice pseudo-inverse de $A$ notée $A^\dagger$ avec $A^\dagger = (A^\top A)^{-1}A^\top$. Le résultat du système est alors $\mathbf{x} = A^\dagger\mathbf{b}$.

# 3 Points et droites à l'infini

## 3.1 Points à l'infini dans $\mathbb{P}^2$

La géométrie dans $\mathbb{P}^2$ correspond à la géométrie du plan en coordonnées homogènes. Voici quelques propriétés des droites et des points du plan utiles pour notre projet sur les points de fuite et les lignes de fuite.

Deux droites parallèles $\mathbf{l_1} = (a, b, c)^\top$ et $\mathbf{l_2} = (a, b, c')^\top$ s'intersectent au point $\mathbf{x} = \mathbf{l_1} \times \mathbf{l_2}$. On a alors $\mathbf{x} = (c - c')(b, -a, 0)^\top \doteq (b, -a, 0)^\top$. La dernière composante de $\mathbf{x}$ étant nulle, $\mathbf{x}$ est un point à l'infini.

L'ensemble des points à l'infini tels que $\mathbf{x}$ reposent sur la droite $\mathbf{l}_\infty = (0, 0, 1)^\top$. En effet, on peut constater qu'un point à l'infini $\mathbf{x} = (x, y, 0)$ vérifie $\mathbf{x}^\top\mathbf{l}_\infty = 0$. On constate aussi que l'intersection d'une droite $\mathbf{l} = (a, b, c)$ avec $\mathbf{l}_\infty$ est le point à l'infini $\mathbf{x} = (b, -a, 0)^\top$ (il en est de même pour $\mathbf{l_2} = (a, b, c')^\top$ parallèle à $\mathbf{l_1}$).

Pour mieux "ressentir" ce qu'est $\mathbf{l}_\infty$, on peut s'imaginer (même si c'est faux) un cercle de rayon infini dont l'origine de notre référentiel est le centre. Deux droites parallèles intersectent ce cercle de part et d'autre en deux points $\mathbf{x} = (b, -a, 0)^\top$ et $\mathbf{x}' = (-b, a, 0)^\top$ (figure 1). En fin de compte, ces deux points représentent le même point à l'infini $\mathbf{x} \doteq \mathbf{x}'$ : l'un de points est le symétrique de l'autre par rapport au centre du cercle.

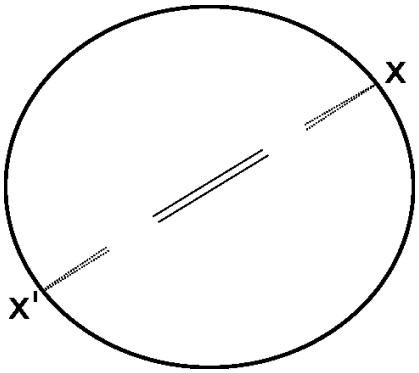


FIG. 1 – Points à l'infini.

### 3.2 Points à l'infini dans $\mathbb{P}^3$

En géométrie projective de l'espace à 3 dimensions, on utilise un outil mathématique nommé plan à l'infini $\pi_\infty$. Il s'agit de l'équivalant de $\mathbf{l}_\infty$ pour un espace de dimension 3. L'une de ses propriétés est que deux plans parallèles $\pi_\mathbf{1}$ et $\pi_\mathbf{2}$ intersectent $\pi_\infty$ sur une même droite : une droite à l'infini. De la même façon, deux droites parallèles $\mathbf{L_1}$ et $\mathbf{L_2}$ intersectent $\pi_\infty$ sur un même point : un point à l'infini.

Là encore, une représentation mentale utile mais incorrecte consiste à imaginer le plan à l'infini $\pi_\infty$ comme une sphère de rayon infini. $\pi_\mathbf{1}$ et $\pi_\mathbf{2}$ intersectent $\pi_\infty$ sur une droite commune qui correspond dans notre représentation à un cercle. De même, $\mathbf{L_1}$ et $\mathbf{L_2}$ intersectent $\pi_\infty$ sur un même point de la sphère (en fait, 2 points dont l'un est le symétrique de l'autre par rapport au centre de la sphère).

## 4 Points de fuite et lignes de fuite

L'étude des points de fuite et des lignes de fuite est couramment utilisée en peinture ou en photographie pour exprimer la perspective. Intuitivement, la perspective en photographie fait que des objets lointains apparaissent plus petits que ces objets s'ils étaient proches. Nous savons aussi que deux droites parallèles dans le monde réel ne le sont pas nécessairement sur une image. Les représentations de ces deux droites sur l'image pointent d'ailleurs vers le même point : un point de fuite (figure 2).

Fig. 2 – Point de fuite

De la même façon, un plan que l'on voit à perte de vue semble avoir une limite, comme l'horizon par exemple : une ligne de fuite (figure 3).

### 4.1 Points de fuite

#### 4.1.1 Définition

Comme il a été précisé plus haut, un point de fuite est associé à une droite et une image. On comprend aisément que le point de fuite $\mathbf{v}$ d'une droite $\mathbf{l}$ est la projection sur l'image du

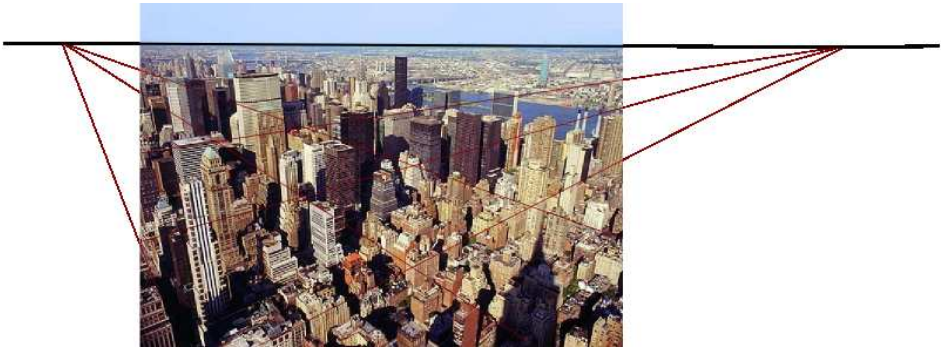

FIG. 3 – Ligne de fuite

point $\mathbf{x} \in \mathbf{l}$ le plus éloigné de la caméra. Pour se rapprocher des notions du paragraphe 3.2, $\mathbf{x}$ correspond au projeté sur l'image du point à l'infini issu de l'intersection de $\mathbf{l}$ avec $\pi_\infty$. En terme de caméra, le point de fuite $\mathbf{v}$ d'une droite $\mathbf{l}$ (de l'espace 3d) est l'intersection du plan image de la caméra avec un rayon parallèle à $\mathbf{l}$ passant par le centre de projection de la caméra.

Une conséquence de ce qui précède est que des droites parallèles au plan image apparaissent parallèles sur l'image. Nous pouvons aussi noter qu'un point de fuite dépend uniquement de l'orientation de la droite à laquelle il est associé, pas de sa position.

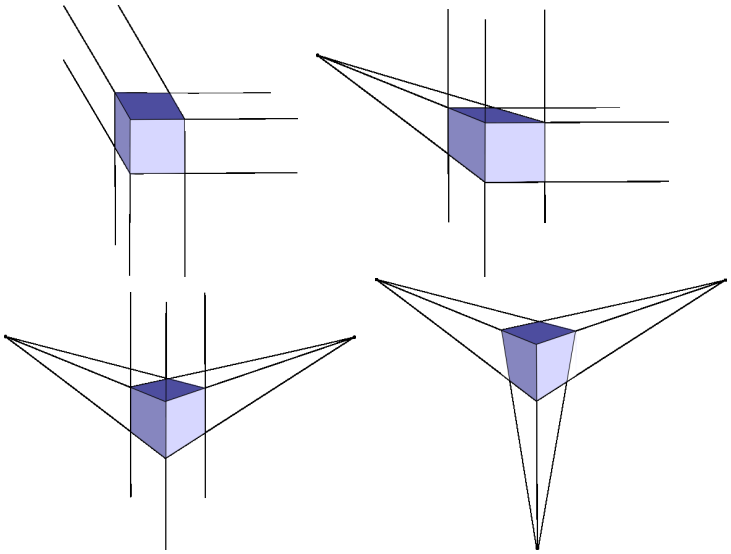

FIG. 4 – nombre de points de fuite variable

### 4.1.2 Calcul d'un point de fuite

En pratique, il existe plusieurs approches permettant de calculer des points de fuite. Le plus simple étant de calculer l'intersection de deux droites de l'image dont on sait qu'elles sont parallèles dans l'espace 3d. Dans le cas de deux droites parallèles $\mathbf{l_1}$ et $\mathbf{l_2}$, leur point de fuite est $\mathbf{v} = \mathbf{l_1} \times \mathbf{l_2}$.

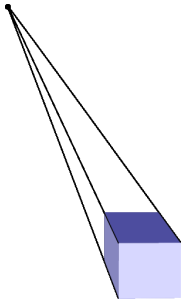

FIG. 5 – Calcul d'un point de fuite

Dans le cas où l'on dispose de $n$ droites parallèles ($n > 2$), il est préférable de toutes les utiliser pour augmenter la précision du calcul de $\mathbf{v}$. On sait que $\mathbf{v}$ appartient à toutes ces droites, ce qui se traduit par $\mathbf{l_1}^\top \mathbf{v} = \mathbf{l_2}^\top \mathbf{v} = \cdots = \mathbf{l_n}^\top \mathbf{v} = 0$. L'ensemble de ces équations peuvent s'assembler sous forme matricielle par le système :

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \vdots & \vdots & \vdots \\ a_n & b_n & c_n \end{bmatrix} \begin{pmatrix} x_v \\ y_v \\ w_v \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Ce système peut se résoudre au sens des moindres carrés en calculant le *right null space* de la matrice des droites à l'aide d'une décomposion en valeurs singulières (SVD). A noter que certaines approches totalement différentes de celle présentée ici sont connues pour avoir de meilleurs résultats. Ces méthodes sont sensibles à la distance entre un segment et son point de fuite présumé.

## 4.2 Lignes de fuite

### 4.2.1 Définition

De même qu'un point de fuite est le point de convergence d'un ensemble de droites parallèles, une ligne de fuite est la droite de convergence d'un ensemble de plans parallèles. On peut aussi voir une ligne de fuite d'un plan comme l'ensemble des points de fuite issus de l'ensemble des droites parallèles appartenant au plan.

Une définition plus formelle consiste à dire que la ligne de fuite $\mathbf{l}$ d'un plan $\pi$ sur une image est le projeté sur cette image de l'intersection de $\pi$ avec le plan à l'infini $\pi_\infty$.

### 4.2.2 Calcul d'une ligne de fuite

On peut définir une ligne de fuite $\mathbf{l}$ d'un plan $\pi$ comme la droite passant par plusieurs points de fuite issus de plusieurs groupes de droites parallèles appartenant à $\pi$. Soient $\mathbf{v_1}$ le point de fuite de deux droites parallèle appartenant à $\pi$ et $\mathbf{v_2}$ le point de fuite de deux autres droites parallèles (entres elles, mais pas avec les deux première droites) appartenant elles aussi à $\pi$. Les points $\mathbf{v_1}$ et $\mathbf{v_2}$ sont tous les deux les projetés des points à l'infini situés sur la droite à l'infini issue de l'intersection de $\pi$ et de $\pi_\infty$. La ligne de fuite $\mathbf{l}$ passe donc par ces deux points et on a alors $\mathbf{l} = \mathbf{v_1} \times \mathbf{v_2}$.

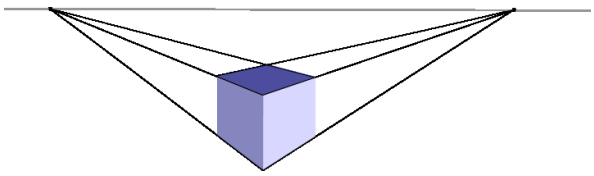

FIG. 6 – Calcul d'une ligne de fuite

# 5 Caméra projective et points de fuites

En vision par ordinateur, une caméra peut être représentée par une matrice de projcetion $P_{3\times4} = K[R|t]$ où $K$ correspond à la matrice $3 \times 3$ des paramètres intrinsèques, $R$ est une matrice de rotation $3 \times 3$ spécifiant l'orientation de la caméra et le vecteur $t$ renseigne sur la position de la caméra. La matrice $K$ se décompose de la manière suivante :

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

avec :

- $f_x$ et $f_y$ la focale selon l'axe des $x$ et l'axe des $y$ (en unité de pixel).
- $s$ le caractère oblique de l'image.
- $c_x$ et $c_y$ l'intersection de l'axe optique avec l'image, exprimé dans le référentiel de l'image.

La caractéristique principale de la matrice de projection $P$ est de projeter un point de la scène $\mathbf{X} = (X, Y, Z, W)^\top$ sur un pixel $\mathbf{x} = (x, y, z, w)^\top$ par $\mathbf{x} = P\mathbf{X}$.

## 5.1 Conique de $\mathbb{P}^2$

Une conique est la courbe définie par l'intersection d'un plan et d'un cône. Dans la famille des coniques, on peut trouver entre autres le cercle, l'ellipse, l'hyperbole et la parabole. Dans $\mathbb{P}^2$, une conique est représentée par une matrice $3 \times 3$ symétrique $C$. Un point $\mathbf{x} \in C$ ssi $\mathbf{x}^\top C\mathbf{x} = 0$.

## 5.2 L'image de la conique absolue : $\omega$

L'image de la conique absolue $\Omega_\infty$, notée $\omega$, est déterminé par l'association du plan à l'infini $\pi_\infty$ et du plan image de la caméra. C'est une notion "imaginaire" et donc non représentable sur une image, cependant $\omega$ est suffisament liée aux paramètres de la caméra pour permettre de calibrer une caméra à partir de quelques informations sur les points ou lignes de fuites.

Tout d'abord, voici la relation qui relie $\omega$ avec la caméra :

$$\omega = (KK^\top)^{-1}$$

La matrice $\omega$ peut se noter de la façon suivante :

$$\omega = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix}$$

A partir de la relation $\omega = (KK^\top)^{-1}$, on peut déduire que si les pixels de l'image sont carrés, alors $f_x = f_y$ et $s = 0$, ce qui se traduit par $\omega_{12} = \omega_{21} = 0$ et $\omega_{11} = \omega_{22}$.
En ajoutant à celà le fait que $\omega$ est symétrique, on obtient une matrice à 4 inconnues :

$$\omega = \begin{bmatrix} \omega_{11} & 0 & \omega_{13} \\ 0 & \omega_{11} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{bmatrix}$$

De plus, les points de fuite $\mathbf{u}$ et $\mathbf{v}$ de deux droites perpendiculaires satisfont $\mathbf{u}^\top \omega \mathbf{v} = 0$. Autrement dit, à partir de trois points de fuites issus de droites ortogonales deux à deux (par exemple : verticale, largeur et profondeur) et du fait que les pixels de nos caméras sont carrés, on peut calculer $\omega$ .

Pour celà, on pose le vecteur $\mathbf{w} = (\omega_{11}, \omega_{13}, \omega_{23}, \omega_{33})^\top$ et on forme une matrice des contraintes $A$ telle que $A\mathbf{w} = \mathbf{0}$. La contrainte $\mathbf{u}^\top \omega \mathbf{v} = 0$ peut s'exprimer à travers $A$ par la ligne :

$$A = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ x_u x_v + y_u y_v & x_u w_v + w_u x_v & y_u w_v + w_u y_v & w_u w_v \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Là encore, ce système peut se résoudre à l'aide d'une SVD.

## 5.3 $K$, $\omega$ et la décomposition de Cholesky

Rappelons que $\omega$ et $K$ sont liés par la relation $\omega = (KK^\top)^{-1}$. Donc une fois $\omega$ trouvé, il ne nous reste plus qu'à calculer $K$ à l'aide de la décomposition de Cholesky.

La décomposition de Cholesky permet de décomposer une matrice symétrique $A$ en $A = LL^\top$ où $L$ est une matrice triangulaire inférieure. En réfléchissant très très fort, on s'aperçoit qu'on

peut constuire $L$ (et par conséquent $L^\top$) en procédant colonnes par colonnes avec les formules suivantes :

$$L_{ii} = \sqrt{(A_{ii} - \sum_{k=0}^{k<i} L_{ik}^2)} \qquad \text{pour } i = j$$

$$L_{ij} = \frac{1}{L_{jj}} \left( A_{ij} - \sum_{k=0}^{k<j} L_{ik} L_{jk} \right) \qquad \text{pour } i > j$$

A noter que le calcul de $K$ n'est pas direct. En effet, puisqu'on a $\omega = (KK^\top)^{-1} = K^{-\top}K^{-1}$, la décomposition de Cholesky nous donne $K^{-1}$ et non $K$.

### 5.4 Position et orientation de la caméra

Une fois la matrice $K$ des paramètres intrinsèques de la caméra trouvée, il ne nous reste plus qu'à "placer" la caméra. Pour cela, il faut définir un referentiel sur l'image spécifiant l'axe des $x$, des $y$ et des $z$ ainsi que l'origine du repère.

L'utilisateur doit alors sélectionner sur l'image les 4 coins d'un carré (dans la scène 3d) dont les coordonnées seront $(0,0,0)^\top$, $(0,1,0)^\top$, $(1,1,0)^\top$ et $(1,0,0)^\top$, c'est à dire un carré de coté 1, sur le plan $z = 0$, orienté vers les $x$ et les $y$, et dont un des coins est sur l'origine du repère (figure 7). En pratique, on n'est pas obligé de choisir un carré, mais ça simplifie les choses.

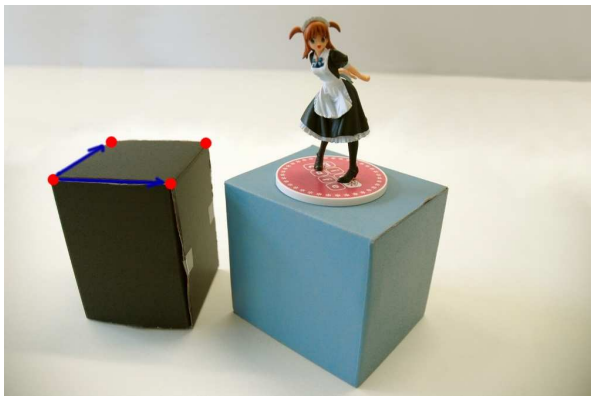

FIG. 7 – Déterminer le repère de scène.

Il faut ensuite calculer l'homographie $H_{3\times3}$ qui transforme les coins 3d du carré (sans prendre en compte leur composante en $z$) avec leur pixels correpondant sur l'image. Quatre points suffisent pour calculer cette homographie (à l'aide d'OpenKraken par exemple).

Concernant notre matrice de projection $P_{3\times4} = K[R|t]$, on sait que la relation $\mathbf{x} = P\mathbf{X}$ est satisfaite pour nos 4 points 3d :

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ w \end{pmatrix} = K \left[\begin{array}{ccc|c} r & r & r & t_x \\ r & r & r & t_y \\ r & r & r & t_z \end{array}\right] \begin{pmatrix} X \\ Y \\ 0 \\ W \end{pmatrix}$$

Ce qui revient à écrire :

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ w \end{pmatrix} = K \left[\begin{array}{cc|c} r & r & t_x \\ r & r & t_y \\ r & r & t_z \end{array}\right] \begin{pmatrix} X \\ Y \\ W \end{pmatrix}$$

On peut constater que cette application transforme en pixels des points 3d privés de leur composante en $z$. Cette application est donc identique à l'homographie $H$ calculée précédemment :

$$K[r_1 r_2|\mathbf{t}] = H$$

soit

$$[r_1 r_2|\mathbf{t}] = K^{-1}H$$

On peut donc par identification trouver $\mathbf{r_1}$, $\mathbf{r_2}$ et $\mathbf{t}$. Sachant que la matrice $R$ est orthogonale (c'est une matrice de rotation), on en déduit $\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}$.

Donc pour finir, on peut calculer notre matrice $P_{3\times4} = K[R|\mathbf{t}]$ en combinant $K$, $R$ et $\mathbf{t}$. A noter que toute cette partie peut elle aussi être effectuée à l'aide de la bibliothèque OpenKraken.

# 6 Réalité augmentée

La réalité augmentée consiste à rajouter de l'image de synthèse sur l'image d'une scène réelle. Pour que les objets calculés en image de synthèse se fondent bien dans l'image, plusieurs critères de compatibilité doivent être satisfaits :

- géométrique entre scène réelle et image de synthèse.
- radiométrique (reflets / ombres / etc. entre objets de synthèse et objet réel).
- "physique du solide" (collision, ...)

Dans notre cas, nous traiterons essentiellement le premier cas en utilisant la matrice de projection $P$ calculée plus haut. A partir d'ici, deux approches sont envisageables : dessiner manuellement sur l'image les projetés de points 3d, ou des segments entre ces projetés (rendu type "fil de fer"). Ou bien un rendu avec OpenGL permettant de dessiner des objets plus complexes, mais nécessitant une adaptation de la matrice de projection $P$ pour qu'elle soit compatible avec les matrices de projection OpenGL.

### 6.1 Sans openGL

Étant donnée une image et sa matrice de projection $P$, dessiner un point $\mathbf{X}$ revient à le projeter sur l'image via la relation $\mathbf{x} = P\mathbf{X}$ et à dessiner le pixel $\mathbf{x}$ avec la couleur voulue. Pour dessiner un maillage, il suffit de trouver les projetés des sommets du maillage et de dessiner les segments entre ces sommets, à l'aide d'OpenKraken par exemple (figure 8).

FIG. 8 – Réalité augmentée sans OpenGL.

### 6.2 Avec openGL

Comme il a été spécifié plus haut, l'utilisation d'OpenGL pour de la réalité augmentée nécessite une adaptation de la matrice de projection $P$ pour que celle-ci soit compatible avec les matrices de projection OpenGL (figure 9). Cette opération peut être réalisée avec la bibliothèque OpenKraken en utilisant les fonctions suivantes :

```
...
float GLProjectionMatrix[16];
float GLModelViewMatrix[16];
...

// camera internal parameters
kn : :ProjectiveCamera camera(K);

// camera external parameters
kn : :Matrix3x3d H = kn : :computeHomography(...);
camera.computeExternalParameters(H);

// OpenGL matrices compatibility
camera.getGLProjectionMatrix(width,height,near,far,GLProjectionMatrix);
camera.getGLModelviewMatrix(GLModelViewMatrix);
...
```

Le rendu se fait alors en 2 phases :

1. l'affichage de l'image en mode `ortho`, sans activer le z-test ni l'eclairage OpenGL.
2. l'affichage des objest 3d en mode perspectif, avec l'éclairage et le z-test activé.

En clair :

```
void display3d(void)
{
  glClear(GL_COLOR_BUFFER_BIT);

  // 2d mode
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0,image.width(), 0, image.height(), -1.0, 1.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glDisable(GL_DEPTH_TEST);
  glDisable(GL_LIGHTING);

  // draw the image with a textured quad
  ...

  // 3d mode
  glMatrixMode(GL_PROJECTION);
  glLoadMatrixf(GLProjectionMatrix);
  glMatrixMode(GL_MODELVIEW);
  glLoadMatrixf(GLModelViewMatrix);
  glEnable(GL_DEPTH_TEST);
  glEnable(GL_LIGHTING);
  glClear(GL_DEPTH_BUFFER_BIT);

  // draw 3d objects
  ...

  glutSwapBuffers();
}
```

# 7 Travail demandé

L'objectif de ce projet est d'implenter un programme d'exploitation des points de fuites d'une image. Ce programme devra satisfaire les conditions suivantes :

- Être écrit en `C++`, utiliser un `makefile` et ne plus afficher de warning lors de la compilation (avec l'option `-Wall`).
- Fonctionner au moins sous Linux (éventuellement portable sous Windows et/ou Mac).

FIG. 9 – Réalité augmentée avec OpenGL.

- Lancement sous console avec la commande suivante : `./executable fichier.ppm [options]`
- Les options suivantes devront fonctionner :
  - `-h` pour afficher l'aide (inspirez vous du man).
  - d'autres options que vous jugerez intéressantes.
- Toutes les options devront être gérées en ligne de commande.
- Ne pas utiliser d'autres interfaces que celles de glut (pas de `cin >> ...`).
- Pouvoir cliquer des correspondances et les sauvegarder dans un fichier, ou pouvoir charger une liste déjà établie. Le format de sauvegarde défini dans la suite de ce document devra être parfaitement respecté.
- Vous devez rendre un rapport sur votre travail.
- Vous devrez rendre votre projet sous forme d'archive `nom1_nom2.tgz` compressant un répertoire du même nom contenant votre programme ainsi qu'un exemple prêt à l'emploi et votre rapport.

Le travail à effectuer sera réparti de la façon suivante :

- La sélection des lignes de fuite via un programme openGL fourni.
- L'extraction des points de fuites et des lignes de fuites.
- Le calcul de l'image de la conique absolue $\omega$.
- La décomposition de Cholesky.
- Le calcul de $K$.
- Le calcul de $H$ via OpenKraken.
- Le calcul de $P$ avec $K$ et $H$ via OpenKraken.
- Un rendu en réalité augmentée.
- Un rapport de 5-10 pages.

Pour chacune de ces étapes, pensez à valider votre travail par des exemples et des images qui figureront dans votre rapport.

### 7.1 Partie informatique

La bibliothèque OpenKraken propose les fonctionalités suivantes :
- La gestion des images PPM.
- La gestion des vecteurs et des matrices
- La résolution de systèmes surdéterminés, le "right null space" de la SVD.
- Le calcul d'inverses de matrices.
- Le calcul d'homographies à partir d'une liste de correspondances.
- Le calcul des paramètres d'une camera à partir de $K$ et $H$.
- La mise en compatibilité des matrices de projection avec OpenGL.

### 7.2 Rapport

Votre rapport devra comporter les points suivants :
- Les méthodes mathématiques qui ne figurent pas dans l'énoncé.
- Les algorithmes utilisés décrits de la façon la plus concise.
- Des résultats (images ou autre).
- Les difficultés rencontrées (d'ordre algorithmique).
- Des idées pour les problèmes non-résolus.
- Une analyse des méthodes numériques utilisées (stabilité ...) si nécessaire.
- Des idées d'améliorations.
- Une liste concise des objectifs atteints, de ce qui ne fonctionne pas et des options ajoutées qui fonctionnent.

## 8 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.