Université Blaise Pascal
U.F.R. Sciences Exactes et Naturelles

Laboratoire des Sciences et Matériaux pour
l'Électronique et l'Automatique
(U.M.R. 6602 du C.N.R.S.)

Mémoire

En vue de l'obtention du
Master STIC : Composants et Systèmes
Spécialité Recherche Vision et Robotique

Un modèle de caméra affine par morceaux

Ludovic Magerand

Encadré par Omar Ait-Aider et Adrien Bartoli

1er octobre 2009

Table des matières

Table des figures

Liste des tableaux

Introduction

Ce mémoire fait suite à un stage de Master spécialisé en recherche vision et robotique. Il présente un modèle de caméra affine par morceaux qui a été étudié au cours de ce travail. Tout en étant plus simple à reconstruire que des modèles comme le modèle perspectif, il devrait améliorer la qualité de la reconstruction par rapport à des modèles plus basiques, à l'instar le modèle affine.

La modélisation des caméras est un domaine très vaste et de nombreuses approches très différentes, donnant naissance à de nombreux modèles, peuvent être utilisées. Le modèle présenté ici étant avant tout basé sur les mathématiques, des notions de bases sur les outils utilisés sont données. Nous ferons aussi un point sur les principaux modèles utilisés à l'heure actuelle dans la reconstruction tridimensionnelle pour les caméras que nous pouvons considérer comme étant standards [1].

Après cette première partie, nous présenterons le modèle mathématique que nous avons élaboré et étudié sous sa forme explicite. Une fois cette première approche réalisée, une illustration géométrique de celui-ci est présentée. Nous verrons après cela que ce modèle sous-tend un deuxième modèle, qui sera plus simple à reconstruire. Nous développerons et illustrerons alors un certain nombre d'ambiguïtés intrinsèques à cette approche, et évoquerons quelques unes des pistes expérimentées pour les résoudre. Après cela nous reviendrons sur l'idée d'origine qui nous a mené à écrire ce modèle en faisant un parallèle avec un des modèles de scènes déformables.

La méthode permettant de reconstruire simplement et rapidement le modèle implicite est alors abordée. La phase de reconstruction du modèle explicite, à partir du modèle sous-jacent, est néanmoins un problème plus difficile et qui nécessite d'imposer un certain nombre de contraintes que nous obtiendrons en manipulant les équations réalisant le lien entre ces deux modèles. Ces manipulations permettent de dégager deux types de contraintes à imposer : contraintes de caméra et contraintes de structure. Ces contraintes imposerons d'utiliser une optimisation sur un critère non-linéaire, qui sera exposé, pour reconstruire le modèle explicite. Il faudra alors initialiser cette optimisation par une valeur dont nous préciserons la nature.

Nous présenterons alors les deux grands types de données que nous avons utilisées pour tester le modèle et son implémentation : données synthétiques et données réelles. Nous montrerons comment nous avons obtenu chacun de ces jeux de données et quelles sont leurs particularités. Après cela la précision des résultats obtenus sera comparée avec celle du modèle affine, et nous aborderons l'utilité de reconstruire ou non l'intégralité du modèle. Nous pourrons enfin regarder le temps de calcul nécessaire à la reconstruction par rapport au modèle affine.

Nous terminerons ce mémoire par une conclusion et une discussion sur le travail effectué jusqu'alors, et ouvrirons le sujet sur quelques pistes de travaux possibles pour améliorer ce modèle.

1. Seront considérées dans ce mémoire comme standards les caméras qui sont facilement accessibles et couramment utilisées, sans système optique exotique ni capteur particulier

1 Préliminaires

Dans cette première section nous allons introduire différents outils qui sont nécessaires à la lecture de ce mémoire. Tout d'abord nous commencerons par définir les notations et conventions qui sont utilisées tout au long de celui-ci. Nous passerons ensuite en revu les notions mathématiques indispensables à la compréhension des équations développées ultérieurement. Nous finirons cette section par une brève présentation des modèles de projection les plus utilisées à l'heure actuelle pour les caméras standards.

1.1 Notations

Nous allons manipuler tout au long de ce mémoire des outils mathématiques, principalement de l'algèbre linéaire. Afin qu'il n'y ait pas de confusion dans les notations et conventions adoptées, nous allons préciser ces dernières :

- Pour ne pas systématiquement devoir transposer les vecteurs, ceux-ci seront représentés par des matrices colonnes. Afin de mieux les identifier dans les équations, ils seront notés en utilisant une police grasse, comme par exemple : $\mathbf{V}$.
- Les matrices, autres que les vecteurs, seront notées en utilisant des lettres majuscules, par exemple : M.
- Les scalaires et indices seront notés en utilisant des lettres minuscules, par exemple : c.
- Les éléments d'une matrice M seront donc naturellement notés : $m_{i,j}$, i étant l'indice de la ligne et j celui de la colonne.
- L'identité matricielle de taille $n \times m$ sera notée : $Id_{n,m}$.

L'intégralité des calculs qui seront développés au cours de ce mémoire relèveront du calcul matriciel. Les opérations standards et élémentaires sur les matrices seront notées généralement selon les conventions habituelles. Soit A et B deux matrices, on notera alors :

- La somme termes à termes de A et B : $A + B$.
- Le produit termes à termes (ou de Hadamar) de A et B : $A \odot B$.
- Le produit standard (ou de Cayley) de A et B : $A \cdot B$.
- L'inverse de A, lorsqu'il existe : A^{-1}.
- La transposée de A : A^T.

De plus, nous manipulerons souvent des séquences d'images composées de pixels qui sont la projection de points tridimensionnels vus par une caméra à un instant donné. Nous avons donc besoin de définir les notations utilisées pour caractériser ou représenter ces différentes entités :

- La séquence d'images sera supposée contenir n images, que l'on nommera aussi vue. Les vues seront indicées par l'indice $i \in [1, n]$.
- Les m points tridimensionnels filmés seront vus comme des vecteurs comportant les trois coordonnées de ce point. Ils seront indicés par $j \in [1, m]$ et notés $\mathbf{Q}_j$.
- La projection de chacun de ces points dans une vue i fixée sera un pixel représenté par un vecteur de deux éléments notés $\mathbf{q}_{i,j}$.
- Lorsque les coordonnées de ces pixels seront centrées par rapport au barycentre des points projetés dans la vue courante, le vecteur sera alors noté $\overline{\mathbf{q}_{i,j}}$.

1.2 Calcul matriciel

1.2.1 Le produit tensoriel

En dehors des opérations standards sur les matrices, nous utiliserons aussi le produit tensoriel (ou de Kronecker), dont nous allons rappeler le principe. Soit A de taille $n \times m$ et B de taille $n' \times m'$, alors le produit tensoriel de A et B sera de taille $n * n' \times m * m'$:

$$A \otimes B = \begin{pmatrix} a_{1,1} \cdot B & \dots & a_{1,m} \cdot B \\ \vdots & \ddots & \vdots \\ a_{n,1} \cdot B & \dots & a_{n,m} \cdot B \end{pmatrix}$$

Ce produit permet donc de construire une matrice constituée de blocs contenant chacun la deuxième matrice pondérée par les coefficients présent dans la première matrice. Dans l'exemple ci-dessus, c'est la matrice B qui est répliquée et pondérée par les coefficients contenus dans A. Il faut noté que ce produit n'est pas commutatif.

1.2.2 La décomposition «RQ»

Elle s'applique à toute matrice M, quelque soit sa taille $n \times m$. La décomposition «QR» factorise la matrice M en deux facteurs tels que l'on ait : $M = Q \cdot R$, où Q est une matrice orthogonale d'ordre n (*i.e.* $Q \cdot Q^T = Id_n$) et R est une matrice triangulaire supérieure de taille $n \times m$. Cette décomposition n'est pas unique et peut être déterminé par plusieurs approches.

Nous préférerons cependant dans ce mémoire l'utilisation de la décomposition équivalente consistant à inverser les deux facteurs : $M = R \cdot Q$, où R est une matrice carré triangulaire supérieur d'ordre n et Q est une matrice de taille $n \times m$ satisfaisant $Q \cdot Q^T = Id$. Cette deuxième version, appelée décomposition «RQ», peut être obtenu à partir de la décomposition «QR» par de simples calculs.

Pour plus de détail sur cette décomposition, et sur les décompositions suivantes, ainsi que des précisions sur leur utilisation dans le cadre de la vision par ordinateur, il est possible de se reporter à [5] par exemple, et en particulier à l'annexe 4. Une autre ressource intéressante quand à l'utilisation de ces décompositions est [8].

1.2.3 La décomposition en valeurs singulières

La décomposition en valeurs singulières d'une matrice réelle permet de résoudre nombre de problèmes très variés, parmi lesquels la détermination du noyau gauche ou droit d'une matrice. Il existe plusieurs manières d'introduire, de démontrer et de calculer mathématiquement cette décomposition bien qu'elles soient équivalentes. Il suffit de se reporter à [4] pour en avoir un exemple.

Considérons l'aspect algébrique, pour une matrice réelle M de taille $n \times m$ donnée, la décomposition en valeur singulière revient à diagonaliser deux matrices symétriques définies positives obtenues en multipliant la matrice M par sa transposée, une fois à droite $(M \cdot M^T)$ et une fois à gauche $(M^T \cdot M)$. Les p valeurs propres non-nulles (réelles et positives) obtenues pour ces deux matrices sont uniques et identiques et permettent de

définir les valeurs singulières de M, c'est à dire un ensemble de valeurs $\{\sigma_i\}_{i\in[1,p]}$ telles que :

$$\forall i \in [1,p], \exists(\mathbf{u},\mathbf{v}) \in \mathbb{R}^m \times \mathbb{R}^n, \begin{cases} M \cdot \mathbf{v} &= \sigma_i \cdot \mathbf{u} \\ M^T \cdot \mathbf{u} &= \sigma_i \cdot \mathbf{v} \end{cases}$$

Les vecteurs $\mathbf{u}$ et $\mathbf{v}$ sont alors les vecteurs singuliers, respectivement à gauche et à droite, de la matrice M. Ils sont respectivement liés aux vecteurs propres de $M \cdot M^T$ et $M^T \cdot M$.

La décomposition en valeurs singulières est alors écrite comme suit :

$$M = U \cdot \Sigma \cdot V^T \tag{1.1}$$

où Σ est l'unique matrice diagonale de taille $n \times m$ obtenue en disposant les valeurs singulières par ordre décroissant sur sa diagonale. Les matrices U et V sont respectivement de taille $n \times n$ et $m \times m$, et formées avec les vecteurs singuliers gauches et droits associés aux valeurs singulières. Ce sont donc des matrices unitaires, mais non uniques.

1.2.4 La décomposition de Cholesky

La décomposition de Cholesky s'applique à toute matrice M carré et qui soit symétrique définie positive. Elle permet de factoriser ce type de matrice en une matrice triangulaire inférieur L et sa transposée :

$$M = L \cdot L^T \tag{1.2}$$

Cette décomposition particulière n'est unique que si lorsque nous imposons que les éléments diagonaux soient tous positifs, c'est la convention généralement adoptée dans les algorithmes qui calculent ce type de décomposition.

1.3 Optimisation

Nous avons besoin dans ce mémoire de faire correspondre un modèle, représenté par une fonction f de n paramètres contenus dans le vecteur $\mathbf{X}$, avec des données dont nous disposons et contenues dans un vecteur $\mathbf{Y}$. Ceci revient mathématiquement à trouver :

$$\min_{\mathbf{X}} \sum_i (y_i - f(\mathbf{X})_i)^2$$

Lorsque la fonction de coût est linéaire, c'est à dire de la forme $f(\mathbf{X}) = A \cdot \mathbf{X}$, où A est une matrice de taille $m \times n$ contenant les m contraintes, nous savons parfaitement résoudre ce type de problème comme en atteste [7] par exemple.

Dans notre cas, nous aurons plutôt à faire à des fonctions qui ne seront pas linéaires. Pour résoudre ce type de problème, il faut utiliser des algorithmes itératifs qui procèdent soit par approximation successive de la solution en linéarisant localement la fonction de coût grâce à son jacobien, c'est ce que fait l'algorithme de Gauss-Newton, soit en calculant le gradient de la fonction de coût pour trouver la direction vers laquelle aller pour minimiser au mieux celle-ci. L'avantage de la méthode de Gauss-Newton est quelle converge plus rapidement vers la solution à partir du moment où elle est proche de celle-ci, alors que la

méthode de descente du gradient converge de façon sûr vers la solution même en en étant éloigné, mais très lentement.

Il existe une méthode qui mélange ces deux approches, il s'agit de l'algorithme de Levenberg-Marquardt. Cette méthode consiste simplement à rajouter aux deux précédentes un coefficient de pondération qui permettra de choisir à chaque itération l'une ou l'autre des méthodes pour faire la nouvelle approximation des paramètres. Ce coefficient est mis à jour à chaque itération en fonction du résultat obtenu pour que la prochaine itération soit plus efficace que l'actuelle.

Comme nous pouvons le deviner, puisque ces méthodes procèdent par itération successives, il faut leurs fournir une solution initiale. Dans le cas de l'algorithme de descente du gradient, tant que celle-ci n'est pas proche d'un minimum local, la convergence vers le minimum est assurée. En revanche, dans le cas de l'algorithme de Gauss-Newton, si la solution initiale est trop éloignée du minimum, la convergence n'est pas certaine. Il convient donc de bien choisir cette solution que l'on fournit au départ car de celle-ci dépend la rapidité et la qualité de la minimisation.

Nous pouvons faire une autre remarque, à chaque itération il faut calculer, ou au moins mettre à jour, le jacobien de la fonction de coût, ce qui implique lorsque nous ne connaissons pas son expression (ce qui est souvent le cas en pratique) de le calculé par une approximation numérique. Lorsque nous avons de nombreux paramètres, ce calcul peut vite devenir très lourd car il demande d'évaluer un grand nombre de fois la fonction de coût uniquement pour cette tâche.

Des précisions supplémentaires et des explications plus détaillées quand à l'utilisation de ces méthodes dans le cadre de la vision par ordinateur sont fournies dans [5] : l'annexe 5 traite de la minimisation de fonctions linéaires, alors que l'annexe 6 traite des méthodes itératives pour la minimisation de fonctions non linéaires.

1.4 Modèles de caméra

Il existe de nombreux modèles pour les caméras standards mais seuls deux grands modèles sont classiquement implémentés et utilisés : le modèle perspectif et le modèle affine. Nous allons les présenter tout les deux succinctement dans cette section.

1.4.1 Modèle perspectif

Le modèle perspectif est obtenu en décrivant mathématiquement et pas à pas le processus de projection qui permet de passer des coordonées des points tridimensionnels projetés aux coordonnées des pixel dans l'image. C'est un modèle qui est très proche de la réalité physique de la projection et qui demande de connaître de nombreux détails physique sur le capteur et le système optique de la caméra : taille des pixels, distance focale, ...

Considérons la figure 1 *p.* 6 et étudions la projection du point de coordonnées homogènes $\mathbf{M}_w$ dans le repère monde. Il faut tout d'abord ramener les coordonnées de ce point dans le repère de la caméra avant de réaliser la projection à proprement parler. Pour cela,

il faut appliquer une translation homogène et une rotation homogène au vecteur $\mathbf{M}_w$:

$$\mathbf{M}_c = \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = T(a,b,c) \cdot R(\alpha,\beta,\gamma) \cdot \mathbf{M}_w \tag{1.3}$$

où a, b et c sont les coordonnées du repère caméra dans le repère monde et α, β et γ sont les angles entre le repère caméra et le repère monde, exprimés selon la convention que l'on souhaite (Euler, «RTL», ...).

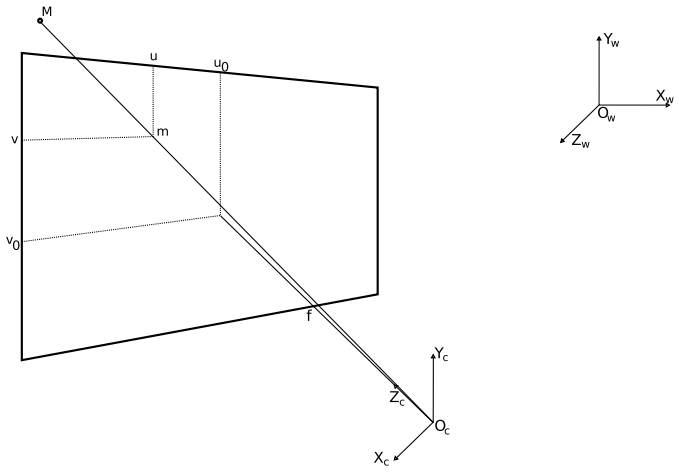

FIGURE 1 – Modèle de projection perspectif.

La projection du point $\mathbf{M}$ en $\mathbf{m}$ sur le plan image de la caméra, situer à la distance focale f de l'origine du repère caméra, se réalise quant à elle à partir des coordonnées dans le repère caméra selon :

$$\mathbf{m}_c = \begin{pmatrix} s \cdot x_i \\ s \cdot y_i \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \mathbf{M}_c \tag{1.4}$$

Il faut maintenant ramener les coordonnées de $\mathbf{m}$, qui sont exprimées dans le repère caméra, dans le repère de l'image. Si nous supposons que le repère image et le repère caméra ont la même orientation, il suffit alors d'appliquer une translation correspondant à la position de la projection de l'origine du repère caméra sur le plan image (u_0 et v_0) et une mise à l'échelle entière correspondant à la taille des pixels (p_x et p_y) :

$$\mathbf{m}_i = \begin{pmatrix} s \cdot u \\ s \cdot v \end{pmatrix} = \begin{pmatrix} \frac{1}{p_x} & 0 & u_0 \\ 0 & \frac{1}{p_y} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \mathbf{m}_c \tag{1.5}$$

Si nous récapitulons l'ensemble de ces opérations, et calculons les produits matriciels des différentes opérations, nous obtenons finalement une matrice de projection de taille 3×4 telle que :

$$\begin{pmatrix} s \cdot u \\ s \cdot v \\ s \end{pmatrix} = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & 1 \end{pmatrix} \cdot \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{1.6}$$

Bien que ce modèle soit le plus complet, nous ne connaissons jamais tout les paramètres internes de la caméra et de la prise de vue qui sont nécessaires pour pouvoir le calibrer. Il faut donc procéder à une estimations des $m_{i,j}$, soit à partir de données obtenues en filmant une scène parfaitement connue (mire) et ne plus modifier les paramètres ensuite, soit en résolvant un système d'équation qui demande une importante quantité de calculs.

Plusieurs méthodes permettent de réaliser la reconstruction du modèle perspectif, nous ne les aborderons pas ici car elles sortent du cadre de ce mémoire, mais il est possible de se référer à [5] pour en avoir un exemple.

1.4.2 Modèle affine

Une excellente présentation des caméras affines est fait dans [9] et une méthode d'auto-calibration de celles-ci y est présentée. Nous avons d'ailleurs utilisé cette méthode pour fournir l'initialisation nécessaire à notre algorithme, comme nous le verrons dans la section 3.4.4 *p.* 24. Reprenons ici quelques éléments qui seront nécessaires dans la suite de ce mémoire.

Une caméra affine réalise une transformation affine de $\mathcal{P}^3$ vers $\mathcal{P}^2$. Si nous reprenons l'équation 1.6 *p.* 7, il suffit de fixer $m_{3,1} = m_{3,2} = m_{3,3} = 0$, contrainte qui a été introduite dans [6]. C'est la forme de caméra affine la plus générale, et différents cas particuliers peuvent être différenciés, mais pour cela, il faut retravailler la matrice de projection, comme cela à par exemple été fait dans [11].

Tout d'abord, nous pouvons constater que $m_{1,4}$ et $m_{2,4}$ correspondent, tout comme dans le cas de la projection perspective, à la translation entre le repère monde et le repère caméra. Si nous nous concentrons alors sur la sous-matrice P_a de taille 2×3 constituée des valeurs restantes, nous pouvons factoriser celle-ci pour faire apparaître une matrice triangulaire supérieure A de taille 2×2 contenant les paramètres dits internes, un peu comme les paramètres internes de la caméra perspective, et une matrice R de taille 2×3 contenant les deux premières lignes d'une rotation qui correspondra à celle entre le repère monde et le repère caméra. Pour cela, nous utilisons la factorisation «RQ» (1.2.2 *p.* 3) :

$$P_a = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \end{pmatrix} = \underbrace{k \cdot \begin{pmatrix} \varepsilon & s \\ 0 & 1 \end{pmatrix}}_{=A} \cdot R \tag{1.7}$$

Les paramètres contenus dans la matrice A n'ont pas une interprétation physique. Nous pouvons tout de même faire un parallèle avec les paramètres de la projection perspective :

– k est comparable à un facteur de mise à l'échelle, dépendant de la taille des pixels du capteur exprimée dans l'unité utilisé pour le repère monde.

– La valeur ε correspond à un ratio d'aspect des pixels lorsqu'ils ne sont pas carrés (sinon $\varepsilon = 1$).
– La dernière valeur (s) est utilisée uniquement lorsqu'il existe un angle non droit entre l'axe vertical et horizontal des pixels du capteur.

Le cas particulier de caméra affine le plus couramment utilisé est la projection orthographique, qui est telle que : $\varepsilon = 1$, $s = 0$ et $k = 1$. C'est ce type de caméra que nous utiliserons plus loin dans ce mémoire.

Il faut noter que le modèle affine permet de reconstruire la scène filmée jusqu'à un facteur d'échelle près qu'il est impossible de déterminer sans un apport d'informations supplémentaire. En effet pour toute solution valide, les coordonnées tridimensionnelles des points peuvent être multipliées par un coefficient fixe quelconque, mais différent de zéro, pour obtenir une solution tout aussi valide.

2 Modélisation affine par morceaux

Nous avons présenté dans la section précédente deux modélisations possibles pour la projection des caméras : perspective et affine. La modélisation affine est très appréciée pour la rapidité qu'elle offre pour la calibration et la reconstruction tridimensionnelle, alors que le modèle perspectif est idéal lorsque l'on souhaite faire une reconstruction plus précise, au détriment du temps de calcul. L'idée développée dans la suite de ce mémoire est de réaliser un modèle projetant les points tridimensionnelles par plusieurs projections affines correspondantes, intuitivement, aux différents plans de la scène. C'est donc une approximation affine par morceaux du modèle perspectif, normalement plus légère à reconstruire et devant permettre un gain en précision par rapport au modèle affine.

Nous pouvons trouver dans la littérature quelques exemples de tels modèles, comme [10], mais ils sont plutôt liés à la reconstruction de surface, ce qui n'est pas le cas du modèle que nous présentons. Nous commencerons par définir explicitement sa forme mathématique puis illustrerons géométriquement le principe de ce modèle. Nous verrons dans un deuxième temps que ce modèle explicite sous-tend un modèle implicite que nous présenterons en détail.

De plus, ce modèle ayant été au départ envisagé comme étant une variation du problème de reconstruction de scènes déformables à partir du modèle de faible rang, très étudié ces dernières années, nous verrons comment il est possible de passer de l'un à l'autre. Ces deux modèles étant d'ailleurs liés, ils partagent un certain nombres d'ambiguïtés, bien que celui présenté ici semble en avoir davantage.

2.1 Modèle mathématique explicite

Les données qui seront traitées par notre modèle prendront la forme de correspondances de points entre les différentes vues d'une séquence vidéo. Un certains nombre d'hypothèses doivent être faites sur ces données et sur la scène filmée dans la séquence afin que le modèle soit le plus simple possible :

- La scène filmée est rigide, ainsi la position tridimensionnelle d'un point entre les différentes vues est fixe.
- Les correspondances entre les points ont été préalablement établies, aucune ne manque et il n'y a pas eu d'erreur d'appariement.

Reprenons les notations définies à la section 1.1 *p.* 2. La projection, dans la vue i, du point $\mathbf{Q}_j$ par la caméra est le pixel de coordonnées $\mathbf{q}_{i,j}$. Pour obtenir un modèle affine par morceaux, cette projection se fait par une combinaison linéaire de l matrices de projection affines $P_{i,k}$, dont les coefficients $\alpha_{k,j}$ dépendent du point projeté. Nous pouvons écrire ceci sous la forme mathématique suivante :

$$\mathbf{q}_{i,j} = \left(\sum_{k=1}^{l} P_{i,k} \cdot \alpha_{k,j} \right) \cdot \mathbf{Q}_j + \mathbf{t}_i \tag{2.8}$$

La détermination de la valeur de l soulève un point important : à quoi correspondent les matrices de projection de bases ? Intuitivement, nous voudrions qu'elles soient liées aux différents plans de la scène. Nous avons donc choisis de déterminer l manuellement en fonction de ce critère. Ainsi pour une scène présentant trois plans par exemple, nous poserons $l = 3$.

Comme il a été vu dans la section 1.4.2 *p.* 7, les caméras affines de base sont données par les matrices $P_{i,k}$, matrices de projection affine de taille 2×3, ainsi que par les vecteurs t_i, vecteurs de translation. Dans la suite de ce mémoire, nous adopterons les notations suivantes pour la décomposition «RQ» de $P_{i,k}$:

$$P_{i,k} = A_{i,k} \cdot R_{i,k} \tag{2.9}$$

où $A_{i,k}$ est la matrice 2×2 triangulaire supérieure, contenant l'équivalent des paramètres internes de la caméra du modèle perspectif, et $R_{i,k}$ est une matrice 2×3 contenant les deux premières lignes d'une matrice orthogonale et représentant la rotation de la caméra.

Si nous reprenons l'équation 2.8 *p.* 9, nous pouvons relativiser les coordonnées des $\mathbf{q}_{i,j}$ dans chaque vue par rapport au barycentre des points de cette vue. Nous éliminons ainsi sans erreurs les translations entre les vues et nous obtenons donc, en notant $\overline{\mathbf{q}_{i,j}}$ les coordonnées centrées :

$$\overline{\mathbf{q}_{i,j}} = \left(\sum_{k=1}^{l} P_{i,k} \cdot \alpha_{k,j} \right) \cdot \mathbf{Q}_j \tag{2.10}$$

2.2 Illustration du modèle

Ce modèle, bien qu'assez intuitif sur le plan mathématique, n'est pas forcément facile à appréhender visuellement. Pour mieux comprendre ce qui ce passe, nous allons consacrer un court instant pour l'illustrer du mieux possible en détaillant la construction géométrique de la figure 2 *p.* 11.

Les points tridimensionnels que l'on projette voient leurs coordonnées exprimées dans un repère monde dont l'origine et l'orientation peuvent varier sans incidence aucune sur le modèle. Posons donc ce repère (en noir sur la figure) et considérons alors que la scène filmée est constituée d'un pavé dont les faces sont peintes de différents niveaux de gris et dont les arrêtes sont noires. Intéressons nous de plus près à la projection de l'un des coins de ce pavé, que nous noterons $\mathbf{Q}_j$ pour conserver les notations de la section 2.1 *p.* 9. Ce point a été marqué en bleu sur la figure.

L'idée du modèle affine par morceaux consiste à projeter ce point avec une combinaison linéaire d'un certain nombre de projection affines. Afin de simplifier l'illustration au maximum, considérons qu'il n'y ait que deux de ces projections, nous en symboliserons une par la couleur verte et l'autre par la couleur rouge. Considérons aussi que nous nous situons dans la i-ème vue de la séquence et nous cherchons donc à construire le point $\mathbf{q}_{i,j}$ qui est la projection de $\mathbf{Q}_j$ dans cette vue.

Dans un premier temps, $\mathbf{Q}_j$ va être transformé en deux points à deux dimensions, qui sont simplement les points obtenus par la projection de $\mathbf{Q}_j$ par chacune des caméras de base $P_{i,1}$ et $P_{i,2}$ pondérées par leurs coefficients respectifs pour ce point : $\alpha_{1,j}$ et $\alpha_{2,j}$. Ces deux points sont donc $P_{i,1}\alpha_{1,j}\mathbf{Q}_j$ et $P_{i,2}\alpha_{2,j}\mathbf{Q}_j$. Le mécanisme de projection affine est symbolisé par les traits en pointillés. Les plans images des deux projections ont été représentés par leurs couleurs respectives, au sein du plan image de la projection de la caméra réelle qui est représentée en noir pour sa part.

Après avoir obtenu ces deux points projetés, il ne nous reste plus qu'à les sommer pour arriver au résultat final, le point $\mathbf{q}_{i,j}$. Ceci est réalisé ici en reportant le vecteur entre

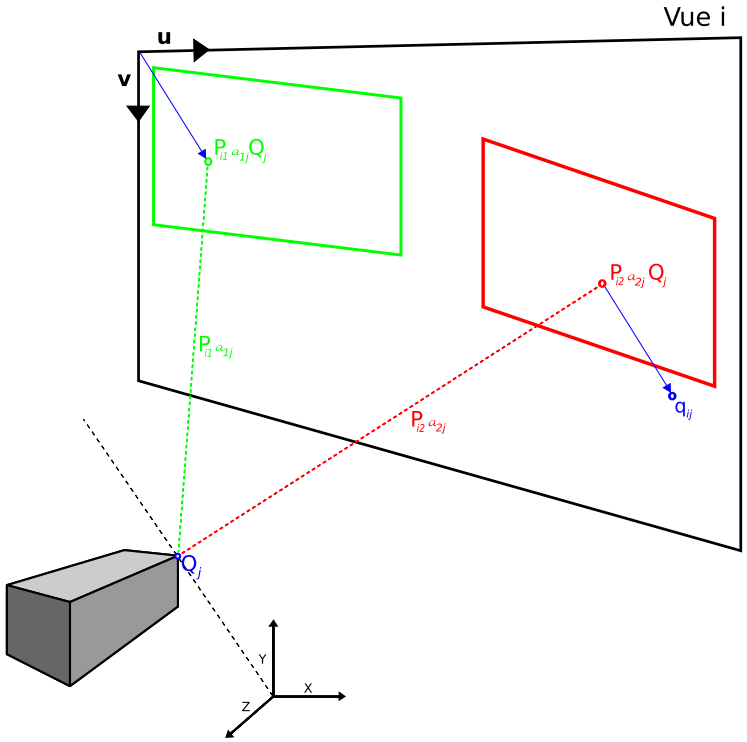

FIGURE 2 – Illustration géométrique du modèle mathématique.

l'origine du plan image et le point $P_{i,1}\alpha_{1,j}\mathbf{Q}_j$ à partir du point $P_{i,2}\alpha_{2,j}\mathbf{Q}_j$. Ce vecteur est représenté en bleu sur la figure.

Notons que bien que nous arrivions au bon résultat, afin de simplifier l'illustration, le mécanisme de projection que nous avons décrit et construit ne correspond pas exactement à ce que nous avons écrit mathématiquement dans la section 2.1 *p.* 9. La différence réside uniquement dans le fait qu'au lieu de sommer les projections affines pondérées de leurs coefficients, ce qui aurait été très difficilement représentable, nous avons sommé les points obtenus après projection par chacune de ces projections. Comme nous allons maintenant le voir dans la section suivante, les deux interprétations sont équivalentes par une simple réécriture des équations de la section 2.1 *p.* 9.

2.3 Modèle implicite sous-jacent

Afin de simplifier le modèle et de le rendre facilement maniable mathématiquement, nous devons remanier les équations qui le décrivent. C'est ce que nous allons nous attacher à faire au cours de cette section. Nous aboutirons ainsi à un deuxième modèle, qui est équivalent à celui-ci mais dont les paramètres auront changé afin de permettre une reconstruction plus facile.

Pour commencer, nous pouvons remarquer que dans l'équation 2.10 *p.* 10, le second membre peut être développé et ensuite récris de la façon suivante :

$$\overline{\mathbf{q}_{i,j}} = P_{i,1}\cdot\alpha_{1,j}\cdot\mathbf{Q}_j + \ldots + P_{i,l}\cdot\alpha_{l,j}\cdot\mathbf{Q}_j \tag{2.11}$$

$$= \underbrace{(P_{i,1}\ldots P_{i,l})}_{=M_i}\cdot\underbrace{\begin{pmatrix}\alpha_{1,j}\cdot\mathbf{Q}_j\\ \vdots \\ \alpha_{l,j}\cdot\mathbf{Q}_j\end{pmatrix}}_{=\mathbf{S}_j} \tag{2.12}$$

Introduisons M_i, matrice de taille $2\times 3l$ et $\mathbf{S}_j$, vecteur de taille $3l$. La matrice M_i contiendra alors les différentes matrices de projection affines de bases pour la vue i, c'est la matrice de projection jointe de cette vue. Le vecteur $\mathbf{S}_j$ est quant à lui un vecteur contenant les cordonnées jointes de chacun des points obtenus par la pondération des coordonnées de $\mathbf{Q}_j$ par les coefficients du modèle.

Nous pouvons maintenant remarquer que le second membre de l'équation 2.11 *p.* 12 est constitué de deux facteurs. Il est donc possible pour toute matrice X carrée de taille $3l$ inversible, de l'introduire avec son inverse entre ces deux facteurs, ce qui donne le résultat suivant :

$$\overline{\mathbf{q}_{i,j}} = \underbrace{M_i\cdot X}_{=J_i}\cdot\underbrace{X^{-1}\cdot\mathbf{S}_j}_{=\mathbf{K}_j} \tag{2.13}$$

Notons J_i, de taille $2\times 3l$, la matrice implicite jointe des caméras de base de la vue i, et $\mathbf{K}_j$, vecteur de taille $3l$, vecteur implicite joint du point $\mathbf{Q}_j$ pondéré de ces coefficients.

C'est la réécriture de notre modèle sous cette forme qui constituera le modèle sous-jacent. Alors que dans le modèle explicite les paramètres étaient constitués directement des informations que l'on connaît où que l'on cherchera à connaître (matrices de projections, coefficients, point tridimensionnel et sa projection), ici ce sont des matrices construites à

partir de ces informations qui constituent les paramètres du modèle. Cette nouvelle forme permettra de reconstruire ces paramètres à la section 3 *p.* 18. En attendant d'aborder ce point, nous allons faire quelques remarques sur ce qu'implique cette réécriture : une ambiguïté.

2.4 Ambiguïtés

Nous avons déjà abordé sans le savoir, dans la section précédente, l'une des nombreuses ambiguïtés que ce modèle peut soulever : celle qui permet d'appliquer les coefficients de pondération soit au point tridimensionnel soit aux matrices de projection. Cette ambiguïté particulière qui est mathématiquement évidente peut parfaitement être illustrée. Nous avons d'ailleurs en partie utilisé cette propriété dans la section 2.2 *p.* 10 car il est plus aisé de représenter sur une figure une combinaison de points, et donc une simple somme de vecteurs, plutôt qu'une combinaisons de projection, et donc de matrices.

Cependant, si nous construisons le modèle obtenu en allant au bout de cette ambiguïté, nous obtenons la figure 3 *p.* 14. Sur celle-ci, nous avons été jusqu'à appliquer les coefficients de pondération directement au point tridimensionnel. Mathématiquement ceci est possible car les coefficients de pondérations sont des scalaires sans dimensions et peuvent donc se déplacer à volonté dans le produit $P_{i,k} \cdot \alpha_{k,j} \cdot \mathbf{Q}_j$.

Si l'on considère les aspects géométrique et physique, cela revient tout simplement à déplacer le point afin de l'amener dans une position où il satisfera au mieux, car il n'est pas complètement libre de se déplacer, les conditions pour pouvoir être projeté par la projection affine correspondante à ce coefficient.

Cette première ambiguïté est la plus évidente, mais elle n'est pas la seule qui existe. Une deuxième ambiguïté apparaît dans le fait que les matrices de projections affines résident dans un espace de dimension finie et que toute combinaison linéaire de matrices qui forment une base de cet espace donne naissance à une nouvelle base. Il se pose donc le problème de déterminer quelle base de matrice de projection affine choisir. Suivant la base choisie, la reconstruction peut être valide ou non, comme on peut le constater dans les figures 4 *p.* 15 et 5 *p.* 16 où la base des matrices de projection affines a été manuellement fixée pour être incorrecte dans la première et correcte dans la seconde. Ces deux reconstructions ont été faite à partir des données présentées section 4.1.2 *p.* 25.

Un autre problème provient du fait que les projections affines contiennent, comme nous l'avons vu à la section 1.4.2 *p.* 7, intrinsèquement une ambiguïté d'échelle. Cette ambiguïté se retrouve dans toute combinaison linéaire de matrices affines, et il n'est pas possible de la résoudre sans un apport d'informations extérieures.

Enfin, une dernière ambiguïté provient du fait que les coefficients peuvent être modifié à volonté, sans forcément invalider l'ensemble du modèle. En effet, ce changement peut simplement s'annuler par passage dans la matrice de projection, ou dans un autre coefficient, via la matrice inversible.

De nombreuses idées ont été testée pour résoudre ces ambiguïtés, certaines avec succès, d'autres moins. Citons en quelques unes :

– Pour choisir la base de projections affines, nous avons appliqué des contraintes ressemblant aux contraintes de base décrites par [12]. Cette solution, qui consiste à fixer les coefficients de certains points choisis manuellement dans les différents plans

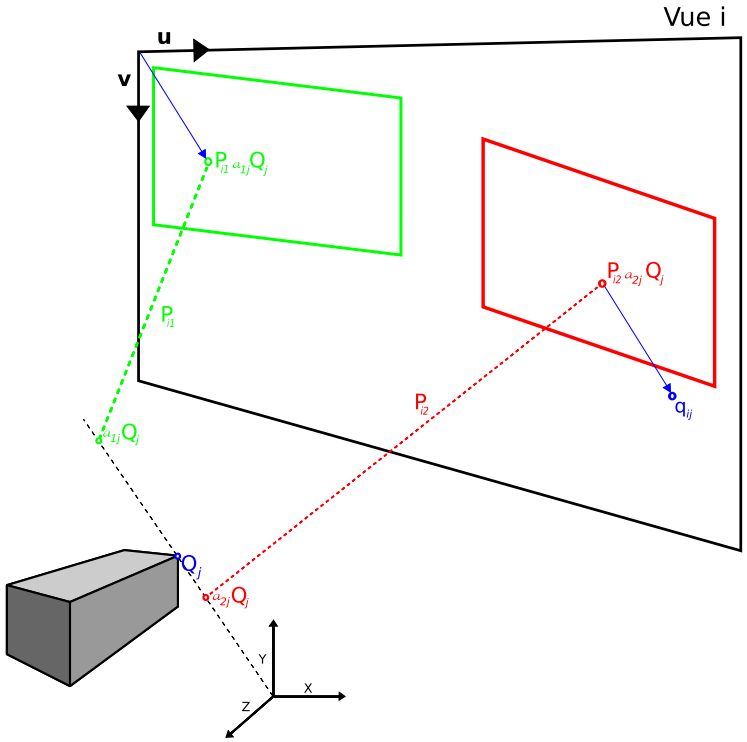

FIGURE 3 – Illustration géométrique d'une des ambiguïtés.

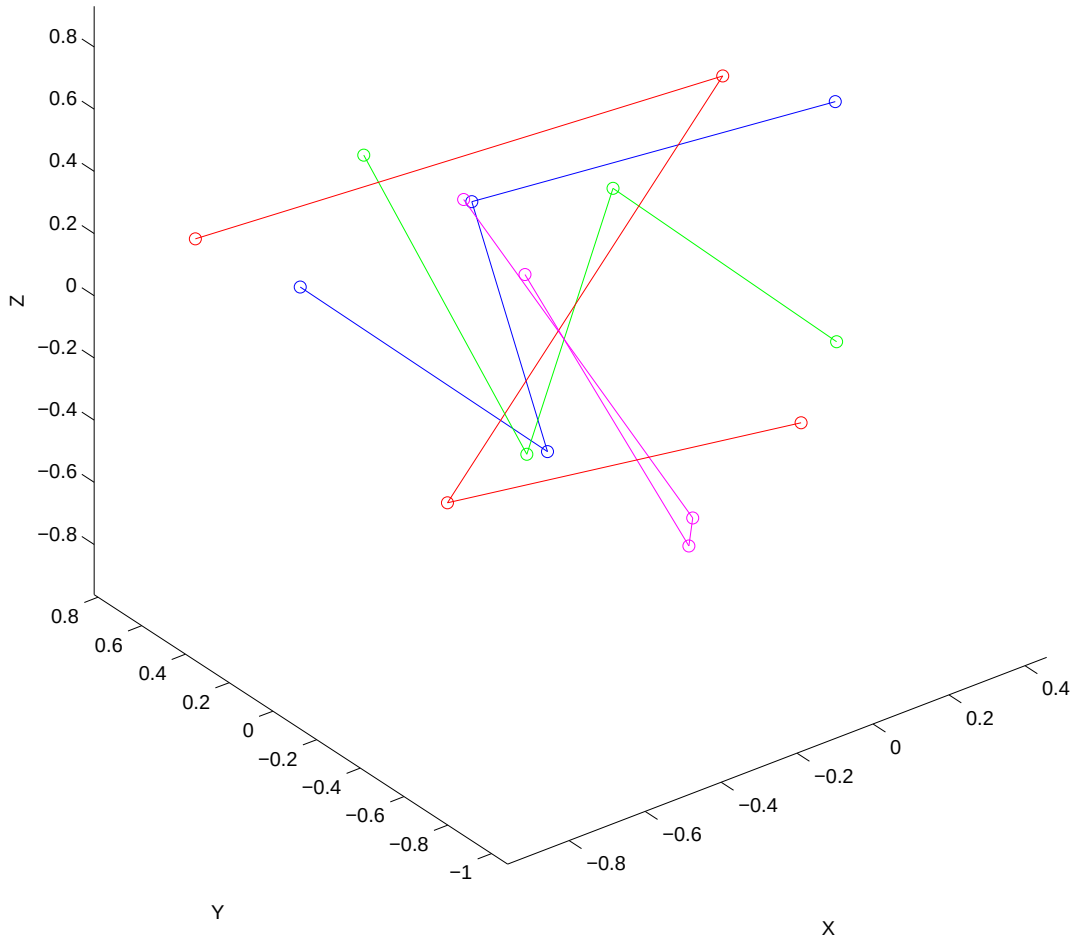

FIGURE 4 – Reconstruction invalide à partir d'un mauvais choix de la base des caméras affines.

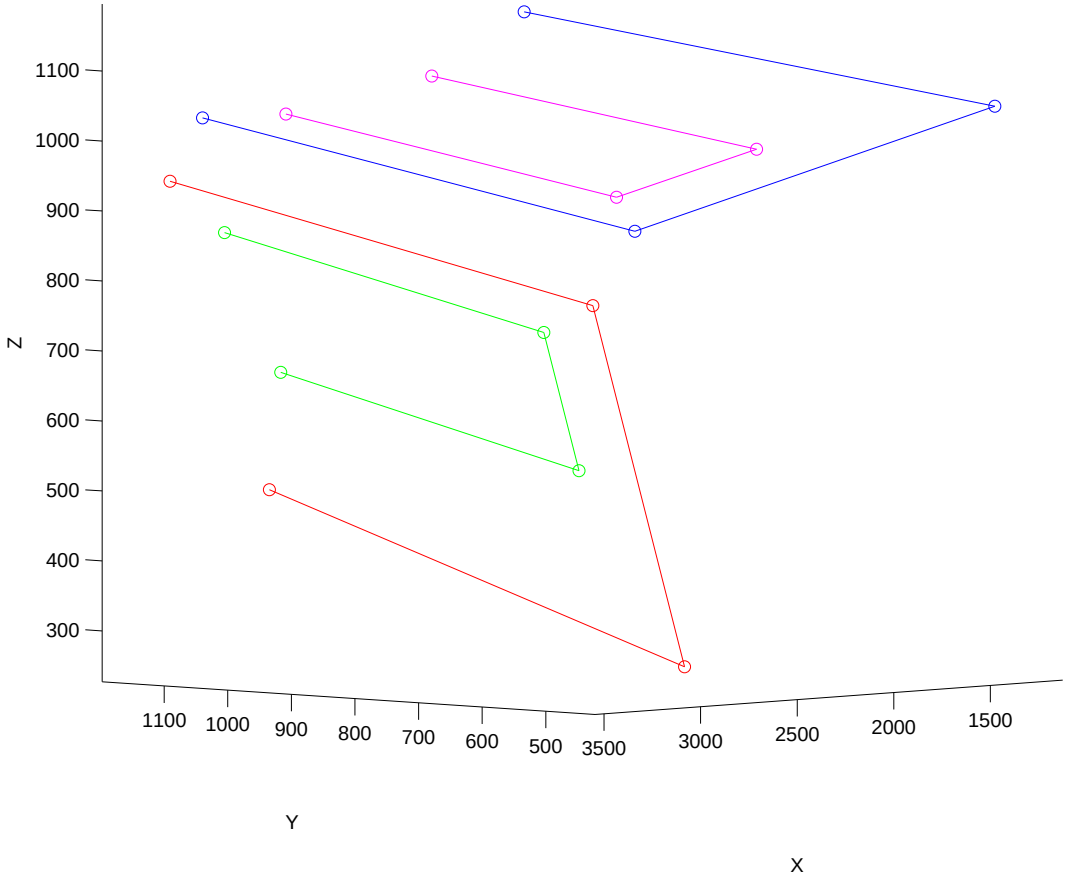

FIGURE 5 – Reconstruction valide après un choix correct de la base des caméras affines.

de la scène, permet de résoudre assez efficacement cette ambiguïté et n'apporte pas de complexité supplémentaire à la reconstruction du modèle.
– Pour essayer de limiter la liberté des coefficients, nous avons tenté d'imposer des contraintes supplémentaires sur ces derniers : leur somme pour chaque point devait être égale à un et ils devaient tous être positifs. L'introduction de ces contraintes a provoquée une importante augmentation du temps de calcul nécessaire pour reconstruire le modèle, et le seule résultat observé a été une convergence systématique de tous les coefficients vers la valeur de $1/l$. Cette idée a donc été abandonnée.

2.5 Lien avec les modèles déformables

Ce modèle présente des similarités avec les modèles décrivant les scènes déformables qui utilisent le modèle de faible rang pour décrire les déformations comme dans [2]. Celles-ci sont en effet alors constituées d'une combinaison linéaire de formes de bases qui sont des points tridimensionnels. Cela revient a écrire exactement le même modèle explicite, à la différence que la combinaison linéaire est déplacé sur les points en lieu et place des matrices de projections :

$$\mathbf{q}_{i,j} = P_i \cdot \left(\sum_{k=1}^{l} \alpha_{i,k} \cdot \mathbf{Q}_{k,j} \right) + \mathbf{t}_i$$

Étant donné que ce type de modèle est étudié depuis plusieurs années et que leur reconstruction a fait d'immenses progrès, ce lien a permis de s'inspirer à plusieurs reprises des solutions trouvées pour ce type de modèle pour les adapter au notre, notamment grâce à [12]. Le même type de méthodes que celles utilisées dans [1] pourraient éventuellement être utilisées pour traiter des données qui ne satisferaient pas certaines des hypothèses que nous avons dû faire, et notamment celle concernant la présence de tout les points d'intérêts dans chacune des images.

3 Reconstruction du modèle

Pour reconstruire le modèle présenté à la section 2 *p.* 9, plusieurs étapes sont nécessaires. Dans un premier temps, grâce à une matrice de mesure construite à partir des données fournies, et que nous présenterons, il est possible de reconstruire le modèle implicite sous-jacent au modèle explicite.

Afin de reconstruire le modèle explicite à partir du modèle implicite, il est nécessaire de retravailler les deux équations qui lient le modèle implicite et le modèle explicite. Ce fessant, il est possible d'extraire de ces deux équations des contraintes de deux types qui permettront de faire évoluer le modèle implicite vers le modèle explicite à l'aide d'une optimisation non-linéaire.

3.1 Construction de la matrice de mesure

Rappelons que nous avons supposé dans la section 2.1 *p.* 9 que les correspondances de points étaient préétablies, manuellement ou à l'aide d'un algorithme automatique robuste, et que ces correspondances étaient donc toutes complètes, c'est à dire présentes dans toutes les images, et sans aucune erreur d'appariement ni de position. Nous pouvons donc former une matrice de mesure complète, que nous nommerons W, sous la forme suivante :

$$W = \begin{pmatrix} \overline{\mathbf{q}_{1,1}} & \dots & \overline{\mathbf{q}_{1,m}} \\ \vdots & & \vdots \\ \overline{\mathbf{q}_{n,1}} & \dots & \overline{\mathbf{q}_{n,m}} \end{pmatrix} \tag{3.14}$$

Si nous reprenons l'équation 2.13 *p.* 12 et que nous substituons les $\overline{\mathbf{q}_{i,j}}$ dans l'équation 3.14 *p.* 18, nous obtenons alors le résultat suivant :

$$W = \begin{pmatrix} J_1 \cdot \mathbf{K}_1 & \dots & J_1 \cdot \mathbf{K}_m \\ \vdots & & \vdots \\ J_n \cdot \mathbf{K}_1 & \dots & J_n \cdot \mathbf{K}_m \end{pmatrix}$$

Dans cette matrice, il est possible de factoriser les J_i sur les lignes et les $\mathbf{K}_j$ sur les colonnes, ce qui revient à récrire la matrice sous la forme :

$$W = \underbrace{\begin{pmatrix} J_1 \\ \vdots \\ J_n \end{pmatrix}}_{=J} \cdot \underbrace{(\mathbf{K}_1 \dots \mathbf{K}_m)}_{=K} \tag{3.15}$$

Introduisons alors les matrices J et K, respectivement de taille $2n \times 3l$ et $3l \times m$, qui seront réutilisées dans la section suivante et qui sont les matrices jointes des caméras à travers toutes les vues, et de tous les points pondérés de leurs coefficients.

3.2 Reconstruction du modèle implicite

La résolution du modèle implicite à partir de la matrice de mesure introduite précédemment est très simple et très rapide. En effet, nous pouvons estimer simplement J et

K à partir de W si cette dernière est au moins de taille $3l \times 3l$, ce qui revient à imposer que les conditions suivantes soient satisfaites :

$$\begin{cases} 2n & > & 3l \\ m & > & 3l \end{cases} \tag{3.16}$$

Lorsque ces deux conditions sont vérifiées, ce qui est généralement le cas car on disposera en principe de bien plus d'images et de points que l'on souhaite avoir de formes de bases, nous pouvons alors effectuer une décomposition en valeur singulière de W, ce qui nous donnera, en reprenant les notations introduites dans la section 1.2.3 *p.* 3 :

$$W = U \cdot \Sigma \cdot V^T \tag{3.17}$$

En ne conservant alors que les $3l$ premières valeurs singulières de cette décomposition et les vecteurs correspondants dans U et V, et en notant Σ', U' et V' les matrices ainsi tronquées, nous pouvons alors choisir de poser :

$$\begin{cases} J & = & U' \\ K & = & \Sigma' \cdot V'^T \end{cases} \tag{3.18}$$

Ce qui donne bien une estimation de J et K. Nous avons signalé lors de l'introduction de la décomposition en valeurs singulières que les matrices U et V n'étaient pas uniques, mais cela à peu d'importance car le modèle implicite tiens déjà compte du fait que la factorisation n'est de toute façon pas unique même si ces deux matrices l'étaient.

3.3 Liens avec le modèle explicite

Après avoir estimé les deux facteurs du modèle implicite, il reste à reconstruire le modèle explicite à partir de ces derniers. Nous pouvons remarquer que les matrices J et K peuvent être récrites, par définition et par l'équation 2.13 *p.* 12, en substituant les matrices jointes de chaque vue et en factorisant la matrice X ou son inverse :

$$\begin{cases} J & = & \underbrace{\begin{pmatrix} P_{1,1} & \dots & P_{1,l} \\ \vdots & & \vdots \\ P_{n,1} & \dots & P_{n,l} \end{pmatrix}}_{=M} \cdot X \\ K & = & X^{-1} \cdot \underbrace{\begin{pmatrix} \alpha_{1,1} \cdot \mathbf{Q}_1 & \dots & \alpha_{1,m} \cdot \mathbf{Q}_m \\ \vdots & & \vdots \\ \alpha_{l,1} \cdot \mathbf{Q}_1 & \dots & \alpha_{l,m} \cdot \mathbf{Q}_m \end{pmatrix}}_{=S} \end{cases}$$

Introduisons alors la matrice M, de taille $2n \times 3l$, matrice jointe explicite de toute les caméras de bases à travers toutes les vues, et la matrice S, de taille $3l \times m$, matrice jointe explicite de la structure et qui combine tous les points tridimensionnels pondérés de tous leurs coefficients.

Nous pouvons alors réarranger successivement les équations du système précédent pour obtenir les égalités suivantes :

$$\begin{cases} J \cdot X^{-1} &= M \\ X \cdot K &= S \end{cases}$$

$$\Rightarrow \begin{cases} {X^{-1}}^T \cdot J^T &= M^T \\ K^T \cdot X^T &= S^T \end{cases}$$

$$\Rightarrow \begin{cases} J \cdot X^{-1} \cdot {X^{-1}}^T \cdot J^T &= M \cdot M^T \\ K^T \cdot X^T \cdot X \cdot K &= S^T \cdot S \end{cases} \tag{3.19}$$

Posons alors $Z = X^T \cdot X$, et remarquons, outre le fait que Z soit symétrique définie positive, que X étant inversible, Z l'est aussi et que nous avons de plus la propriété suivante sur son inverse :

$$\begin{aligned} Z^{-1} &= (X^T \cdot X)^{-1} \\ &= X^{-1} \cdot {X^T}^{-1} \\ &= X^{-1} \cdot {X^{-1}}^T \end{aligned}$$

Ceci implique que nous pouvons alors récrire le système d'équations 3.19 *p.* 20 ainsi :

$$\begin{cases} J \cdot Z^{-1} \cdot J^T &= M \cdot M^T \\ K^T \cdot Z \cdot K &= S^T \cdot S \end{cases} \tag{3.20}$$

3.4 Évolution vers le modèle explicite

La réécriture du système d'équations 3.20 *p.* 20 sous cette forme va permettre de faire plus facilement sortir deux types de contraintes que nous allons pouvoir chercher à minimiser afin de reconstruire le modèle explicite à partir des facteurs J et K du modèle implicite. De plus, cette dernière forme fait intervenir une matrice Z qui sera plus aisée à paramétrer que la matrice X.

3.4.1 Paramétrisation de Cholesky

La matrice Z introduite à la fin de la section 3.3 *p.* 19 a, par définition, la propriété, qu'il faudra maintenir, d'être symétrique et définie positive. Par conséquent, en invoquant le théorème de la décomposition de Cholesky (*c.f.* 1.2.4 *p.* 4), nous pouvons paramétrer cette matrice en utilisant une matrice triangulaire inférieure L de même taille que Z :

$$Z = L \cdot L^T \tag{3.21}$$

La contrainte de symétrie est naturellement maintenue par cette paramétrisation, il ne reste plus qu'à maintenir la matrice Z définie positive. Ceci peut être accompli en imposant que le dernier terme de la diagonale de L soit égal à un. En effet, multiplier L par une

constante revient simplement à multiplier Z par le carré de cette constante, et comme ce terme ne peut être égal à zéro puisque la matrice est définie positive et donc non singulière, nous pouvons imposer cette contrainte sans perte de généralité sur la paramétrisation.

Cette paramétrisation, bien que non linéaire, est très simple à mettre en oeuvre et à utiliser. De plus, elle conserve le nombre de paramètres indépendant de la matrice Z :

$$\frac{3l * (3l + 1)}{2} - 1 \tag{3.22}$$

3.4.2 Contraintes de caméra

Revenons maintenant au système d'équation 3.19 *p.* 20, et plus particulièrement à la première ligne. Puisque cette ligne fait apparaître les matrices jointes implicite et explicite des caméras, nous l'appellerons l'équation des contraintes de caméra. Intéressons nous plus particulièrement au deuxième terme de cette égalité, que nous pouvons par définition écrire comme ceci :

$$\begin{aligned} MM^T &= \begin{pmatrix} P_{1,1} & \dots & P_{1,l} \\ \vdots & \ddots & \vdots \\ P_{n,1} & \dots & P_{n,l} \end{pmatrix} \cdot \begin{pmatrix} P_{1,1}^T & \dots & P_{n,1}^T \\ \vdots & \ddots & \vdots \\ P_{1,l}^T & \dots & P_{n,l}^T \end{pmatrix} \\ &= \begin{pmatrix} P_{1,1}P_{1,1}^T + \dots + P_{1,l}P_{1,l}^T & \dots & \\ \vdots & \ddots & \vdots \\ & \dots & P_{n,1}P_{n,1}^T + \dots + P_{n,l}P_{n,l}^T \end{pmatrix} \end{aligned}$$

Rappelons que les matrices $P_{i,k}$ sont des matrices de projections affines, et que par conséquent, comme vu à la section 1.4.2 *p.* 7, elles sont décomposables en une matrice de taille 2×2 notée $A_{i,k}$ et une matrice 2×3 noté $R_{i,k}$. Cette dernière étant composée des deux premières lignes d'une matrice orthogonale, nous avons donc la propriété suivante :

$$\begin{aligned} P_{i,k} \cdot P_{i,k}^T &= A_{i,k} \cdot R_{i,k} \cdot R_{i,k}^T \cdot A_{i,k}^T \\ &= A_{i,k} \cdot A_{i,k}^T \end{aligned}$$

Cette propriété implique que nous pouvons récrire la matrice $M \cdot M^T$ selon :

$$MM^T = \begin{pmatrix} A_{1,1}A_{1,1}^T + \dots + A_{1,l}A_{1,l}^T & \dots & \\ \vdots & \ddots & \vdots \\ & \dots & A_{n,1}A_{n,1}^T + \dots + A_{n,l}A_{n,l}^T \end{pmatrix} \tag{3.23}$$

Intéressons nous maintenant au premier terme de l'égalité de l'équation matricielle des contraintes de caméra, nous pouvons aussi le récrire :

$$JZ^{-1}J^T = \begin{pmatrix} J_1Z^{-1}J_1^T & \dots & J_1Z^{-1}J_n^T \\ \vdots & \ddots & \vdots \\ J_nZ^{-1}J_1^T & \dots & J_nZ^{-1}J_n^T \end{pmatrix} \tag{3.24}$$

Si nous considérons l'égalité entre les deux expressions précédente, donnée par l'équation 3.19 *p.* 20, et en particulier l'égalité sur les bandes diagonales, nous pouvons alors obtenir le système suivant :

$$\begin{cases} J_1 Z^{-1} J_1^T &= A_{1,1}A_{1,1}^T + \ldots + A_{1,l}A_{1,l}^T \\ &\vdots \\ J_n Z^{-1} J_n^T &= A_{n,1}A_{n,1}^T + \ldots + A_{n,l}A_{n,l}^T \end{cases} \tag{3.25}$$

Jusque ici nous avons travaillé avec des matrices de projection de bases qui sont sous la forme affine la plus générale, en effet aucune hypothèse n'a été faite sur les paramètres de la caméra. Plusieurs hypothèses peuvent être faite pour simplifier l'écriture précédente, chacune apportant des types de contraintes différentes. Plusieurs de ces hypothèses ont été testées et nous pouvons en citer deux qui sont les plus intéressantes :

- Les paramètres internes des matrices de projection de bases restent, pour une caméra de base donnée, constant à un facteur d'échelle près à travers les vues. L'avantage de cette hypothèse réside dans le fait que ainsi ces caméras de bases sont des caméras affines les plus générales possible, ce qui semble être une bonne chose pour la qualité du modèle. Néanmoins, elle aboutit à des équations qui, outre le fait qu'elles soient non linéaires, demandent un important travail de combinatoire pour toutes les expliciter alors même qu'un grand nombre d'entres elles sont redondantes. Le coût de ce calcul ne justifie pas le gain obtenu, et c'est pourquoi cette solution n'a pas été retenue et que la solution suivante lui a été préférée.
- Les caméras de base sont toutes des caméras orthographiques, c'est à dire avec des pixels carrés : ratio d'aspect égal à un et l'axe vertical est orthogonal à l'axe horizontal. Cette hypothèse peut être considérée comme valide car les caméras standards la satisfont généralement. Lorsque ce n'est pas le cas, il est de toutes façons facile de corriger l'image pour se ramener à une situation proche de cette hypothèse. Cela revient à écrire que nous avons la forme suivante pour la matrice $A_{i,k}$:

$$A_{i,k} = c_{i,k} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.26}$$

$$\Rightarrow A_{i,k} \cdot A_{i,k}^T = c_{i,k}^2 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.27}$$

Si nous introduisons cette dernière hypothèse dans l'équation 3.25 *p.* 22, nous obtenons alors le nouveau système suivant :

$$\begin{cases} J_1 Z^{-1} J_1^T &= (c_{1,1} + \ldots + c_{1,l}) \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ &\vdots \\ J_n Z^{-1} J_n^T &= (c_{n,1} + \ldots + c_{n,l}) \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{cases} \tag{3.28}$$

Nous obtenons ainsi n égalités entre des matrices de taille 2×2, qui sont toutes deux symétriques par définition. Nous pouvons donc obtenir deux contraintes pour chaque équation de ce système :

– Une première, obtenue en utilisant une des deux valeurs hors-diagonales,
– Une deuxième, obtenue en annulant l'effet des facteurs d'échelle en effectuant le rapport des deux valeurs diagonales :

$$\frac{j_k Z^{-1} j_k^T}{j'_k Z^{-1} j'^T_k} = 1 \tag{3.29}$$

en notant j_k et j'_k les deux lignes de J_k.

Ces contraintes ressemblent beaucoup aux contraintes métriques utilisées pour l'auto-calibration des caméras affines, à l'exception qu'ici un facteur d'échelle, constitué de la somme des facteurs d'échelles des l caméras de bases, apparaît dans le système 3.28 *p.* 22.

3.4.3 Contraintes de structure

Nous allons maintenant nous intéresser à la deuxième ligne du système 3.19 *p.* 20. Celle-ci fait pour sa part intervenir les matrices jointes implicites et explicites de structure, il s'agira donc tout naturellement de l'équation des contraintes de structure.

Là aussi, il est possible de récrire le deuxième terme de cette équation, pour faire apparaître cette fois-ci des produits particuliers :

$$S^T \cdot S = \begin{pmatrix} \alpha_{1,1} \cdot \mathbf{Q}_1^T & \dots & \alpha_{l,1} \cdot \mathbf{Q}_1^T \\ \vdots & & \vdots \\ \alpha_{1,m} \cdot \mathbf{Q}_m^T & \dots & \alpha_{l,m} \cdot \mathbf{Q}_m^T \end{pmatrix} \cdot \begin{pmatrix} \alpha_{1,1} \cdot \mathbf{Q}_1 & \dots & \alpha_{1,m} \cdot \mathbf{Q}_m \\ \vdots & & \vdots \\ \alpha_{l,1} \cdot \mathbf{Q}_1 & \dots & \alpha_{l,m} \cdot \mathbf{Q}_m \end{pmatrix}$$

$$= \begin{pmatrix} (\alpha_{1,1}^2 + \ldots + \alpha_{l,1}^2)\mathbf{Q}_1^T\mathbf{Q}_1 & \dots & (\alpha_{1,1}\alpha_{1,m} + \ldots + \alpha_{l,1}\alpha_{l,m})\mathbf{Q}_1^T\mathbf{Q}_m \\ \vdots & \ddots & \vdots \\ (\alpha_{1,1}\alpha_{1,m} + \ldots + \alpha_{l,1}\alpha_{l,m})\mathbf{Q}_m^T\mathbf{Q}_1 & \dots & (\alpha_{1,m}^2 + \ldots + \alpha_{l,m}^2)\mathbf{Q}_m^T\mathbf{Q}_m \end{pmatrix}$$

Nous pouvons en effet remarquer la présence de nombreux produit du type $\mathbf{Q}_j^T\mathbf{Q}_{j'}$. Ces produits sont des produits vectoriels entre les deux vecteurs formés en reliant l'origine du repère monde à chacun des points $\mathbf{Q}_j$ et $\mathbf{Q}_{j'}$. Sur la diagonale, nous pouvons même constater qu'il s'agit en fait du carré de la distance entre le point et l'origine du repère monde.

Si nous nous intéressons maintenant au premier terme de l'équation des contraintes de structure, nous pouvons nous apercevoir qu'il est de la même forme que celui de l'équation des contraintes de caméra, et peut donc être récris de la même manière :

$$K^T Z K = \begin{pmatrix} K_1^T Z K_1 & \dots & K_1^T Z K_m \\ \vdots & \ddots & \vdots \\ K_m^T Z K_1 & \dots & K_m^T Z K_m \end{pmatrix} \tag{3.30}$$

Ces deux termes étant symétriques, il est donc inutile de considérer toute la matrice pour écrire les contraintes de structure. Nous conserverons donc uniquement la partie triangulaire inférieure, ce qui nous donne l'ensemble d'équations :

$$\begin{cases} \forall j \in [1,m], & K_j^T Z K_j = (\alpha_{1,j}^2 + \ldots + \alpha_{l,j}^2)\mathbf{Q}_j^T\mathbf{Q}_j \\ \forall (j,j') \in [1,m] \times [1,m]/j \neq j', & K_j^T Z K_{j'} = (\alpha_{1,j}\alpha_{1,j'} + \ldots + \alpha_{l,j}\alpha_{l,j'})\mathbf{Q}_j^T\mathbf{Q}_{j'} \end{cases} \tag{3.31}$$

Nous obtenons ainsi $\frac{m*(m+1)}{2}$ contraintes de structure. Celles-ci peuvent être directement écrites à l'intérieur d'une seule matrice à partir de matrices contenant les inconnues (coefficients et points). Notons C la matrice de taille $l \times m$ qui contiendrait les coefficients $\alpha_{k,j}$ et Q la matrice de taille $3 \times m$ qui contiendrait les points $\mathbf{Q}_j$. Nous avons alors la propriété suivante :

$$K^T \cdot Z \cdot K = \left(\left(C \otimes \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}\right) \odot \left(Q \otimes \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}\right)\right)^T \cdot \left(\left(C \otimes \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}\right) \odot \left(Q \otimes \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}\right)\right)$$

3.4.4 Critère à minimiser et initialisation

Pour faire la reconstruction vers le modèle explicite, nous allons minimiser la somme des erreurs au carré sur les contraintes que nous avons calculées dans les deux sections précédentes, et qui sont fonction des variables suivantes :

- Les valeurs de la matrice L vue dans la section 3.4.1 *p.* 20 et utilisées pour construire la matrice Z,
- Les coefficients de pondérations, qui seront contenus dans la matrice C,
- Les coordonnées des points, qui seront contenus dans la matrice Q.

Étant donné que ces deux critères ne sont pas linéaires, nous utiliserons la méthode de Levenberg-Marquardt présentée à la section 1.3 *p.* 4 pour minimiser cette somme. Il nous faudra donc trouver une valeur initiale pour les différents paramètres. Nous avons adoptés les valeurs suivantes :

- La matrice L est égale à l'identité,
- Les coefficients de pondérations sont tous égaux à un,
- Les coordonnées des points sont égales à celles obtenues avec une calibration affine simple.

Nous avons choisi d'initialiser les points en utilisant une calibration affine simple car elle fournit normalement une reconstruction qui bien que inadaptée à une caméra perspective, ne devrait pas se trouver très loin de la solution réelle. Néanmoins, comme nous le verrons dans la section 4.2 *p.* 28, cette solution initiale semble être problématique et n'est peut être pas la meilleure.

4 Résultats

Tout au long de l'élaboration de la méthode de résolution, les différentes étapes ont été implémentées afin de les valider et de vérifier le fonctionnement de l'algorithme. Deux types de données ont été utilisées pour cela, nous allons les présenter.

Nous comparerons ensuite la précision de notre modèle, lorsque la reconstruction est réussie, avec celle du modèle affine. Les possibilités d'exploitations des matrices de projections reconstruites et des coefficients associés seront aussi abordées. Pour finir, nous nous intéresserons au temps de calcul nécessaire pour reconstruire notre modèle, tant dans son évolution en fonction des implémentations que dans sa comparaison avec le modèle affine.

4.1 Données utilisées pour les tests

Les données utilisées pour tester le modèle sont de deux types : des données synthétiques générées automatiquement en simulant une caméra, et des données issues de séquences d'images acquises à l'aide d'une caméra réelle. Nous allons présenter la méthode d'obtention de ces deux types de données.

4.1.1 Génération de données synthétiques

Les données synthétiques ont été obtenues en implémentant une caméra artificielle qui suit le modèle de projection perspectif développé dans la section 1.4.1 *p.* 5. Nous avons ainsi pu obtenir des données exactes et satisfaisant les hypothèses que nous avons faites, ainsi que des jeux de données imparfaites dont on contrôlait le bruit.

Deux objets virtuels ont été imaginés :

- Un pavé de $1m \times 1m \times 2m$ comme sur la figure 6 *p.* 26,
- Les trois faces d'un pavé de $0,5m \times 0,3m \times 0,7m$ recouvert partiellement d'un maillage de $0,1m$ visible figure 7 *p.* 26.

Les figures 6 à 8 montrent quelques images que nous pouvons construire à partir des données synthétiques. Nous pouvons remarquer sur la figure 8 que notre caméra virtuelle ne tient pas du tout compte des occultations possibles de l'objet par lui-même, ce qui est voulu puisque cela permet de tester l'algorithme en respectant les hypothèses de départ, en particulier concernant la présence de tous les points dans toutes les vues. Cependant, les effets perspectifs de la caméra sont bien présents et évidents, voir même exagérés, bien que les paramètres internes de la caméra virtuelle aient été choisis à partir de caméras réelles calibrées.

4.1.2 Acquisition de données réelles

L'autre type de données a été obtenu en réalisant l'acquisition avec une caméra réelle, dont on ne connaissait pas les paramètres internes, de scènes rigides constituées d'éléments comportant des points d'intérêts facilement identifiables avec une bonne précision. Ces points ont été mis en correspondances manuellement afin de s'assurer qu'il n'y avait pas d'erreurs d'appariement ou de positionnement.

Les figures 9 *p.* 27 et 10 *p.* 27 montrent quelques unes des images issues de ces acquisitions, et les points d'intérêts utilisés ont été identifiés sur la figure 10 *p.* 27.

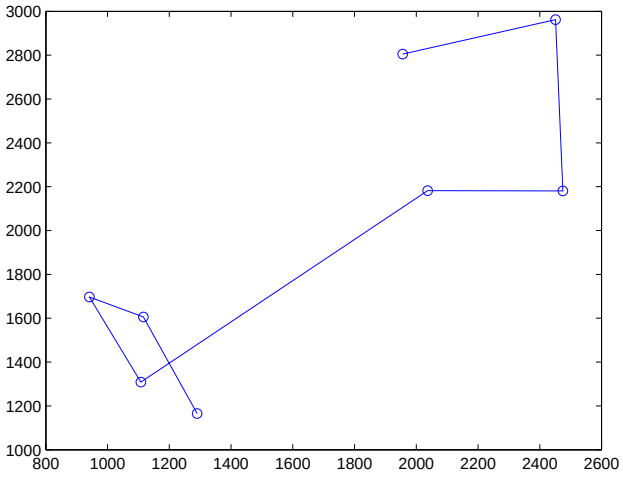

FIGURE 6 – Exemple de données synthétiques, cas du pavé simple.

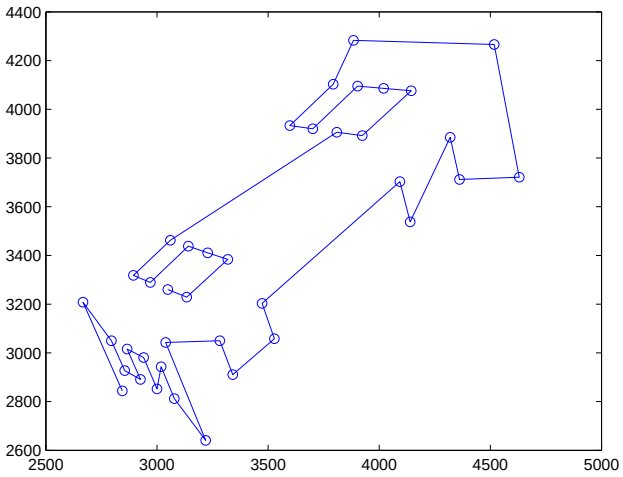

FIGURE 7 – Exemple de données synthétiques, cas du pavé maillé.

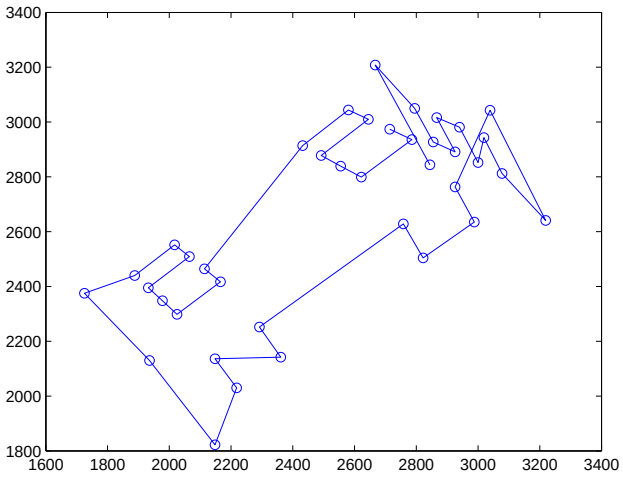

FIGURE 8 – Exemple de données synthétiques, cas d'occultation.

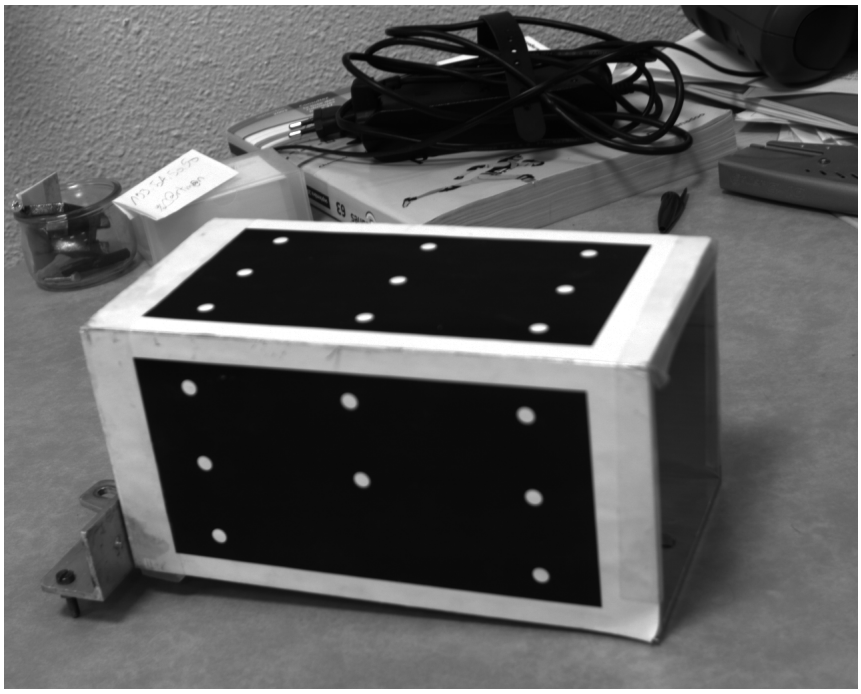

FIGURE 9 – Exemple de données réelles acquises.

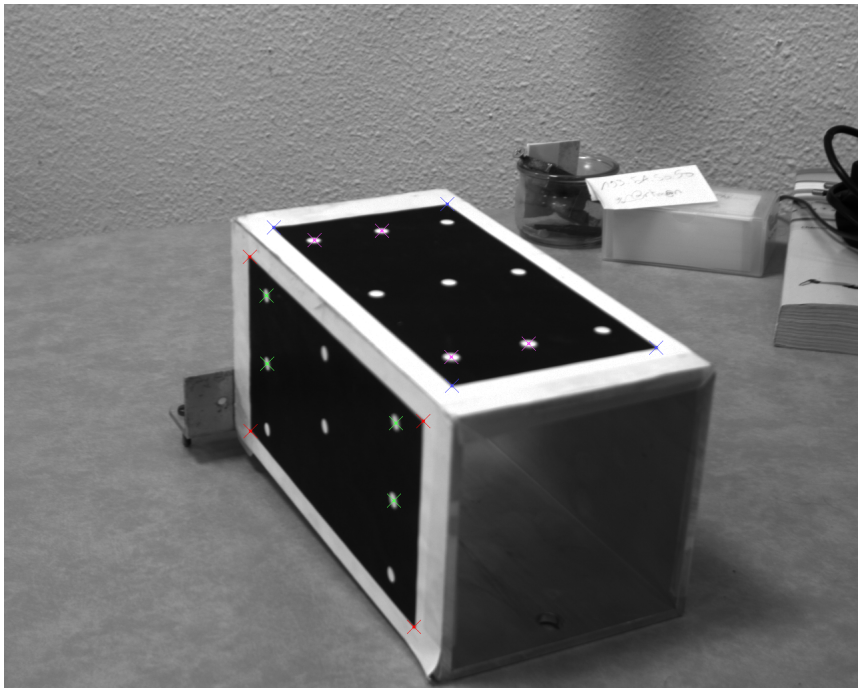

FIGURE 10 – Exemple de données réelles acquises avec les points d'intérêts.

4.2 Précision comparée au modèle affine

Nous pourrions penser que cette modélisation devrait apporter un gain en précision assez important, néanmoins pour différentes raisons, le gain par rapport à la reconstruction affine simple n'est que de quelques pour cents. Ce résultat a été obtenu en relevant les angles entre des arêtes particulières de nos données simulées dont nous savons qu'elles forment des angles droits. Nous avons comparé l'erreur commise sur l'angle reconstruit entre notre modèle et le modèle affine. La diminution d'erreur obtenue sur la reconstruction de ces angles n'est que de deux à trois pour cents. Pour confirmer ce résultat, nous avons aussi comparé les distances entre des points dont nous savions qu'ils étaient équidistants. L'erreur commise sur ces distances est inférieure de quatre à cinq pour cents à celle obtenue pour le modèle affine.

Cet étrange résultat n'est peut être pas si étonnant que cela. En effet, la décomposition en valeur singulière classe les vecteurs singuliers en fonction des valeurs singulières, qui sont classées par ordre d'importance, ce qui signifie que les vecteurs singuliers qui correspondent aux premières matrices de projection sont prépondérantes par rapport aux autres. C'est d'ailleurs ce qui fait que la reconstruction affine simple fonctionne car si on se limite à la première matrice de projection, nous avons celle qui est prépondérante. Ajouter les matrices suivantes n'est donc peut être pas pertinent si la différence entre les valeurs singulières est trop importante, ce que nous avons constaté assez souvent tant sur les données simulées que sur les données réelles.

Une deuxième explication possible provient de la solution initiale choisie pour la minimisation du critère de reconstruction. En effet, nous avons choisi de prendre les points tridimensionnels obtenus par une reconstruction affine simple, or l'expression des contraintes de caméra que nous avons obtenus à la section 3.4.2 *p.* 21 est très proche de celle obtenu pour l'auto-calibration de caméras affine standards.

Des tests supplémentaires et plus complets, utilisant de nouveaux jeux de données réelles et simulées, seraient nécessaires pour approfondir l'étude et déterminer les raisons exactes à l'origine de ce gain en précision si faible. Néanmoins, le principe de la méthode reste validé puisque, aussi faible qu'il soit, le gain est présent.

4.3 Exploitation des matrices de projection et des coefficients

La reconstruction de notre modèle nous fourni les informations suivantes : les coordonnées des points tridimensionnels, les coefficients de pondération, la matrice inversible X sous la forme de la matrice Z, et les matrices de projection sous la forme de la matrice J. Les coordonnées des points sont directement exploitables et constituent l'information probablement la plus importante puisqu'il s'agit d'informations directes sur la scène filmée. Néanmoins, nous devons nous interroger sur les possibilités d'utiliser les autres informations.

Plusieurs pistes d'exploitation de ces données peuvent être envisagées. Commençons par les matrices de projection : grâce à elles il serait éventuellement possible de rajouter ou modifier la scène en projetant de nouveaux points sur la scène d'origine ou en modifiant les points existants. Cependant, pour recouvrer les matrices de projection à partir de la méthode de résolution précédente, il faudrait commencer par retrouver la matrice inversible X à partir de la matrice Z. Il s'agit d'un problème apparemment simple, mais ce n'est pas

le cas. En effet, nous avons $X^T \cdot X = Z = L \cdot L^T$, ce qui implique que $X = L \cdot O$, où O est une matrice orthogonale. Malheureusement, le recouvrement de la matrice O n'est pas une tâche évidente. De plus, l'intérêt serait quand même très limité, car pour pouvoir projeter de nouveaux points correctement dans l'image, il faudrait pouvoir trouver les coefficients de pondération qui leurs seraient liés, ce qui n'est pas une tâche facile non plus.

La deuxième piste possible concerne les coefficients, il serait en effet tentant de penser que ceux-ci devraient permettre de segmenter la scène en fonction de leurs valeurs si l'on considère que l'algorithme devrait tendre à utiliser chacune des matrices de projection affine pour chacun des plans. Néanmoins, les résultats expérimentaux tendent à montrer que ce n'est pas ce qui ce passe réellement et que le phénomène est plus complexe. Si nous observons la table 1 *p.* 29, qui contient une partie des coefficients obtenus après une reconstruction avec deux matrices de bases d'une scène simulée, nous pouvons constater qu'ils tendent à être faibles et à privilégier généralement la première matrice de projection. Ce dernier point tend d'ailleurs à confirmer la remarque faite à la section 4.2 *p.* 28. Cependant, dans de rares cas nous observons que les coefficients privilégient la deuxième matrice de projection, bien que nous classerions probablement pas manuellement les points correspondants ensemble dans un même plan de la scène. Il faudrait néanmoins tester l'utilisation d'un algorithme de segmentation robuste comme celui présenté dans [3] pour voir si il ne se dégagerait pas un phénomène intéressant. De plus lorsque nous augmentons le nombre de matrice de bases, cette analyse devient difficile, comme nous pouvons le constater dans la table 2 *p.* 29.

1.0000	0	0.0044	0.0038	0.0030	0.0040	0.0114	0.0018	0.0030
0	1.0000	0.0021	-0.0005	0.0022	0.0038	-0.0106	0.0040	0.0061
0.0043	0.0016	-0.0009	0.0051	0.0034	0.0055	0.0039	0.0053	0.0008
0.0067	0.0058	0.0067	0.0073	0.0027	0.0052	0.0029	0.0048	0.0050

TABLE 1 – Exemple de coefficients obtenus après une reconstruction à deux matrices de bases.

0.0032	0.0056	0.0051	0.0052	0.0064	0.0033	0.0050	0.0037	0.0053
0.0053	0.0034	0.0050	0.0070	0.0030	0.0063	0.0035	0.0070	0.0035
0.0053	0.0055	0.0045	0.0039	0.0066	0.0056	0.0071	0.0046	0.0064

TABLE 2 – Exemple de coefficients obtenus après une reconstruction à trois matrices de bases.

Il semblerait donc que les données autres que les points soient plus ou moins inexploitables en l'état, c'est pourquoi nous avons privilégié pour l'instant une méthode de reconstruction du modèle qui n'impose pas un recouvrement facile de ces informations.

4.4 Temps de calcul comparé au modèle affine

La première remarque que nous pouvons faire dés à présent et au vu de la section 3.4.4 *p.* 24, c'est que puisque nous initialisons la minimisation du critère de reconstruction de notre modèle par les points tridimensionnels obtenus à partir d'une calibration affine standard, nous mettrons indéniablement plus de temps à reconstruire notre modèle qu'il

n'en faut pour reconstruire le modèle affine simple. Ce point ce vérifie dans la figure 11 *p.* 30 où l'on peut constater que la fonction qui réalise la calibration affine simple (*pacm_solve_affine*) s'exécute en une seconde environ alors que le temps total utilisé pour l'ensemble de l'algorithme est d'un peu plus de douze secondes. Ce surcoût n'est pas négligeable et ne peut se justifier que si la reconstruction obtenu ainsi est vraiment plus efficace qu'une simple calibration affine.

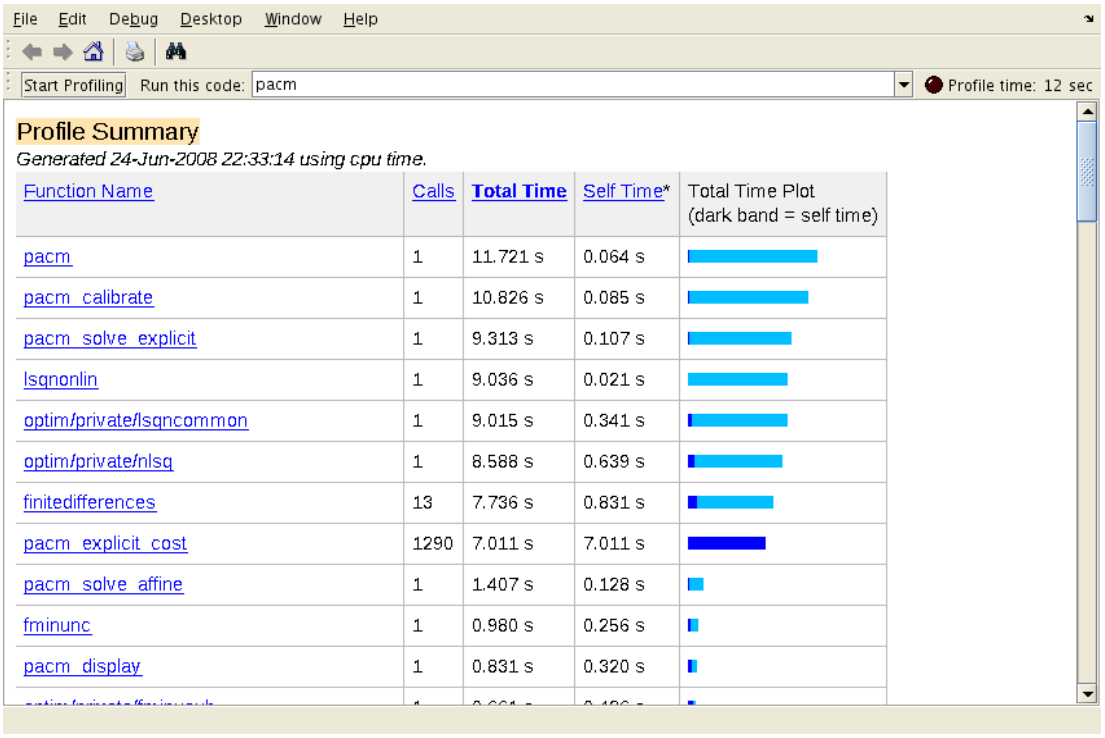

FIGURE 11 – Parts du temps de calculs utilisé dans la dernière implémentation.

Ensuite la deuxième remarque à faire, et que nous pouvons constater en comparant la figure 11 *p.* 30 et la figure 12 *p.* 31 est que le temps de calcul dépend énormément de la façon dont est réalisé l'implémentation de la minimisation et notamment du calcul de la fonction de coût, ce qui est prévisible car elle est évalué de très nombreuses fois pour le calcul du jacobien. Lors de l'implémentation de la méthode, de nombreuses reprises ont été nécessaire avant d'arriver à obtenir une implémentation de la fonction de coût qui soit relativement efficace, ou du moins qui semble l'être. Les temps de calculs sont ainsi passés de plusieurs dizaines de minutes avec l'implémentation d'origine à quelques minutes maximum seulement dans la dernière version implémentée.

Un bon moyen d'améliorer le temps de calcul serait de réalisé une linéarisation des contraintes utilisés pour reconstruire le modèle, en imposant de nouvelles hypothèses ou restriction sur celui-ci.

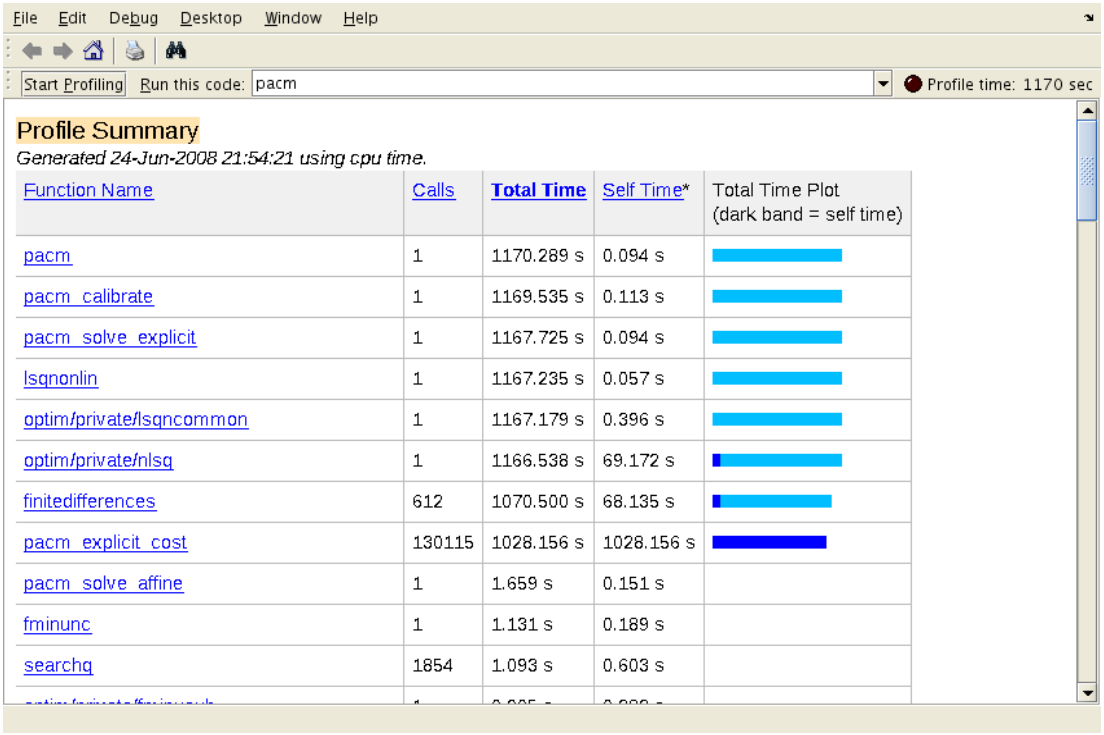

FIGURE 12 – Parts du temps de calculs utilisé dans les premières implémentations.

Conclusion et discussion

Au cours de ce mémoire, nous avons présenté, dans le but d'améliorer la reconstruction tridimensionnelle fournie par le modèle affine standard, un modèle de projection que nous pouvons qualifier d'affine par morceaux de part sa forme mathématique et l'interprétation qui en est possible. Nous avons ensuite illustré le principe de fonctionnement de ce modèle en construisant pas à pas la projection d'un point dans une vue. Nous nous sommes alors intéressé au modèle implicite sous-jacent et après cela nous avons exposé quelques unes des ambiguïtés qui sont propres à cette approche. Nous les avons illustrées à l'aide de figures issues de reconstruction où ces ambiguïtés n'étaient pas résolues.

Nous nous sommes ensuite attaché à montrer comment il est possible de reconstruire ce modèle, en commençant par la reconstruction, très simple, du modèle sous-jacent. Pour retrouver le modèle explicite à partir de ce résultat nous avons développé, à partir des équations liants les modèles implicite et explicite, des contraintes de caméra et de structure qui forment un critère à minimiser et qu'il faut initialiser.

Finalement, nous avons montré que ce modèle peut fonctionner en utilisant des données synthétiques et réelles. Nous avons constaté une petite amélioration de la précision de la reconstruction par rapport à ce qui est obtenu avec un modèle affine classique. Néanmoins ce gain en précision doit être relativisé en prenant en compte le surcoût calculatoire qu'il induit. De plus, nous avons montré qu'en dehors des coordonnées reconstruites pour les points tridimensionnels, les paramètres restants du modèle sont inexploitables sans ajouter encore de lourds calculs.

Nous sommes donc en droit de nous interroger sur l'intérêt de ce modèle, malgré le fait qu'il existe de nombreuses pistes pour l'améliorer. Nous pouvons citer à titre d'exemple la possibilité d'initialiser les coefficients de pondération en réalisant une segmentation de la scène à partir d'une technique basée sur le flot optique entre les premières images. Il serait aussi possible d'envisager le modèle sous une forme stratifiée, c'est à dire qu'au lieu de le reconstruire en une seule fois, le nombre de formes de bases serait augmenté itérations par itérations.

Références

[1] A. E. Bartoli and S. I. Olsen. A batch algorithm for implicit non-rigid shape and motion recovery. In *Workshop on Dynamical Vision*, pages 257–269, 2006.

[2] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages II : 690–696, 2000.

[3] D. Comaniciu and P. Meer. Mean shift : A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(5) :603–619, May 2002.

[4] G. H. Golub and C. F. Van Loan. *Matrix Computation*. John Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, Maryland, third edition, 1996.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, June 2004.

[6] J. L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[7] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, second edition, 2006.

[8] N. Paragios, Y. Chen, and O. D. Faugeras. *Handbook of Mathematical Methods in Computer Vision*. Springer, 2006.

[9] L. Quan. Self-calibration of an affine camera from multiple views. *International Journal of Computer Vision*, 19(1) :93–105, July 1996.

[10] M. Salzmann, V. Lepetit, and P. Fua. Deformable surface tracking ambiguities. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages 1–8, 2007.

[11] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography : A factorization method. *International Journal of Computer Vision*, 9(2) :137–154, Nov. 1992.

[12] J. Xiao, J. X. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. *International Journal of Computer Vision*, 67(2) :233–246, Apr. 2006.