École Normale Supérieure Paris-Saclay

Centre de Mathématiques et de Leurs Applications

Rapport de stage de licence

Bundle adjustment with known positions

Antoine BARRIER
Ségolène MARTIN

Encadrants : Carlo DE FRANCHIS
Jean-Michel MOREL

25 Janvier 2017 — 29 Juin 2017

Introduction

Contexte et situation

La modélisation en 3 dimensions à partir d'images réalisées par une ou plusieurs caméra(s) est un problème dont les applications se multiplient avec le développement de nouvelles technologies : cartographie 3D, estimation des dégâts lors d'une catastrophe naturelle, estimation de la fonte des glaciers ...

La technique de modélisation 3D la plus couramment employée s'appuie sur le principe de la **vision stéréoscopique** : les relations entre deux photographies d'une même scène rigide prises depuis deux points de vue différents permettent de reconstituer le relief de la scène.

Le problème du Bundle Adjustment

Cependant, pour calculer les coordonnées 3D d'un point identifié sur plusieurs images, il est nécessaire de connaître les **positions** et les **orientations** des caméras. Ce n'est en général pas le cas, ou alors à une incertitude près qui ne permet pas d'avoir une reconstitution satifaisante.

On essaye donc d'estimer positions et coordonnées à partir de correspondances entre les images. Cette méthode, connue sous le nom de **bundle adjustment**, permet en théorie d'estimer une reconstruction 3D de la scène modulo une similitude de l'espace. Cependant, les algorithmes d'optimisation actuels ne semblent pas fonctionner totalement : s'ils permettent bien de remodéliser la scène 3D, c'est en fait plus du à une compensation d'erreurs qu'à une estimation exacte des paramètres de la caméra. Nous cherchons donc à estimer de manière exacte ces paramètres.

Pour remédier à cela, nous avons travaillé sur un cas particulier du problème : celui où les positions des caméras sont connues. Cette hypothèse permet effectivement de simplifier la phase de bundle adjustment et semble de plus amplement justifiée : de nombreux systèmes (voitures, drones, robots, avions) embarquent des caméras munies d'un système type GPS qui permet de connaître les positions des caméras avec une précision suffisante (de l'ordre du cm). C'est alors la mesure de l'orientation qui est la plus sensible, car suivant la distance entre la caméra et la scène, une erreur angulaire même très faible peut induire une erreur importante sur la position des points 3D (un milli-radian à 100 mètres de distance donne une erreur de 10 cm).

Cette situation semble aussi particulièrement adaptée aux images satelittes, les positions des caméras étant connues avec une forte précision puisque les trajectoires sont programmées à l'avance. Résoudre ce problème pourrait donc permettre une cartographie 3D précise de la Terre.

Objectifs

L'objectif de ce stage est d'étudier le problème du bundle adjustment dans le cas où les positions des caméras sont supposées connues précisément. Le but étant de **trouver le nombre minimal de vues nécessaires pour obtenir une reconstruction 3D de la scène à partir de mesures faites uniquement sur les images** (c'est-à-dire des correspondances entre images obtenues de façon entièrement automatique avec des algorithmes comme la méthode SIFT dont la précision est de l'ordre de 0.1 pixel), avec une précision contrôlée. On pourra alors programmer un algorithme capable de retourner les paramètres des caméras associées à différentes prises de vues avec une précision contrôlée en ne connaissant qu'une estimation de ces paramètres ainsi que des points de concordance entre images.

Travail

Au cours de ce stage, nous avons eu différentes tâches à réaliser :

- un travail bibliographique pour assimiler les connaissances sur le sujet,
- une identification des difficultés de notre problème, mais aussi des simplifications apportées par les hypothèses dans lesquelles nous nous plaçons,
- une implémentation en Python de nos méthodes, mais aussi des méthodes de bundle adjustment classique, afin de les comparer,
- différentes simulations sur des données simulées puis sur des données réelles,
-

Remerciements

Plan du rapport

1 Description du problème

1.1 Rappels sur les caméras

On modélise une caméra par la donnée d'une matrice de taille 3×4 de la forme $\mathrm{P} = \mathrm{KR}[\mathrm{I}_3 \mid -\mathrm{C}]$ où :

- K contient les paramètres internes de la caméra : distance focale, résolution, informations sur le repère de l'image ...,
- R modélise l'orientation de la caméra : c'est une matrice de rotation,
- C est la position du centre de la caméra.

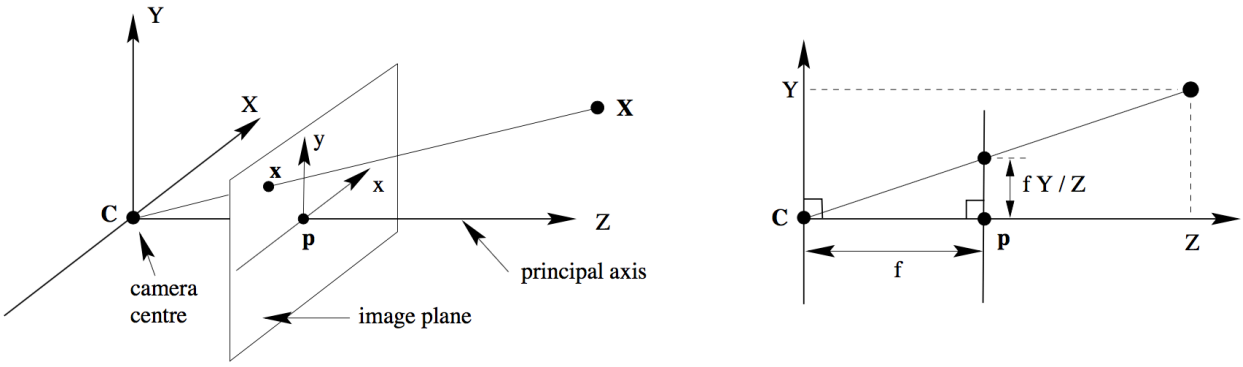

Figure 1: Modélisation d'une caméra **[changer notations figure ...]**

Pour des descriptions plus détaillées, on pourra se référer au chapitre 6 du livre de Hartley et Zisserman [1].

1.2 Situation et notations

On se place dans le référentiel terrestre noté $\mathbf{R}_T = (O, X_T, Y_T, Z_T)$.

On s'intéresse à K images prises par des caméras à K positions distinctes **connues précisément** pour lesquelles on a N points de correspondance (grâce à la méthode SIFT ...).

On note $\mathbf{R}_{cam\ i} = (\mathrm{C}_i, X_{cam\ i}, Y_{cam\ i}, Z_{cam\ i})$ le repère de la i-ième caméra, $\mathbf{R}_{im\ i}$ le repère du plan image de la i-ième caméra et on suppose précisément connues les coordonnées du centre C_i de chaque caméra.

On suppose de plus que les matrices K_i des paramètres internes des caméras sont toutes égales à

$$\mathrm{K} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Pour des questions de compatibilité avec nos codes Python, on numérotera les caméras de 0 à $K-1$ et les points de 0 à $N-1$.

On dispose également :

- d'une **estimation** – pas assez précise – des angles de rotation des caméras, que l'on regroupe selon le vecteur :
$$\theta_0 = (\tilde{\alpha}_0, \tilde{\beta}_0, \tilde{\gamma}_0, \dots, \tilde{\alpha}_{K-1}, \tilde{\beta}_{K-1}, \tilde{\gamma}_{K-1})^\top$$
où $\tilde{\alpha}_i$, $\tilde{\beta}_i$ et $\tilde{\gamma}_i$ sont les angles de rotation estimés de la i-ième caméra respectivement par rapport à X_T, Y_T et Z_T. Ainsi, la matrice de rotation de la i-ième caméra se décompose sous la forme :
$$\mathrm{R}_i = \begin{pmatrix} \cos(\tilde{\gamma}_i) & -\sin(\tilde{\gamma}_i) & 0 \\ \sin(\tilde{\gamma}_i) & \cos(\tilde{\gamma}_i) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\tilde{\beta}_i) & 0 & \sin(\tilde{\beta}_i) \\ 0 & 1 & 0 \\ -\sin(\tilde{\beta}_i) & 0 & \cos(\tilde{\beta}_i) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\tilde{\alpha}_i) & -\sin(\tilde{\alpha}_i) \\ 0 & \sin(\tilde{\alpha}_i) & \cos(\tilde{\alpha}_i) \end{pmatrix}$$
Remarque L'ordre des matrices et donc des trois rotations sous-entend que l'on considère des rotations définies de manière extrinsèque.
- d'une **estimation** – là encore pas assez précise[1] – des coordonnées des images sur chaque caméra des points pour lesquels on a une correspondance, regroupés selon le vecteur :
$$\mathbf{X}_0 = (\underbrace{\tilde{x}_0^0, \tilde{y}_0^0, \dots, \tilde{x}_i^0, \tilde{y}_i^0, \dots, \tilde{x}_{K-1}^0, \tilde{y}_{K-1}^0}_{\text{coord. des images du point 0}}, \dots, \underbrace{\tilde{x}_0^j, \tilde{y}_0^j, \dots, \tilde{x}_{K-1}^j, \tilde{y}_{K-1}^j}_{\text{coord. des images du point } j}, \dots, \underbrace{\tilde{x}_0^{N-1}, \tilde{y}_0^{N-1}, \dots, \tilde{x}_{K-1}^{N-1}, \tilde{y}_{K-1}^{N-1}}_{\text{coord. des images du point } N-1})^\top$$
où $(\tilde{x}_i^j, \tilde{y}_i^j)^\top$ sont les coordonnées estimées de l'image du j-ième point sur la i-ième caméra (donc exprimées dans le repère $\mathrm{R}_{im\ i}$).

1.3 Démarche

Notre problème consiste à estimer avec la meilleure précision possible les valeurs réelles θ_* des angles ainsi que $\mathbf{X}_*$ des coordonnées des points que l'on regroupera dans des vecteurs.

Pour cela, on dispose des valeurs de θ_0 et de $\mathbf{X}_0$ ainsi que des conditions imposées par les correspondances des points, à savoir le fait que pour tout $0 \leqslant j \leqslant N-1$ fixé, on sait que les droites $(\mathrm{C}_i \mathbf{x}_i^j)$ sont toutes sécantes en un même point (qui est le j-ième point de correspondance mais dont on ne connaît pas les coordonnées).

Ces conditions vont nous permettre de voir notre estimation $(\theta_a, \mathbf{X}_a)$ de $(\theta_*, \mathbf{X}_*)$ comme la solution d'une équation du type $F(\theta, \mathbf{X}) = 0$ au voisinage de $(\theta_0, \mathbf{X}_0)$.

[1] les algorithmes de correspondance n'ont pas une précision parfaite

2 Formules associées au problème

2.1 Deux caméras et un point

On considère ici le cas où $K = 2$ et $N = 1$. [2]

On a $\theta_0 = (\tilde{\alpha}_1, \tilde{\beta}_1, \tilde{\gamma}_1, \tilde{\alpha}_2, \tilde{\beta}_2, \tilde{\gamma}_2)^\top$ et $\mathbf{X}_0 = (\tilde{x}_1, \tilde{y}_1, \tilde{x}_2, \tilde{y}_2)^\top$.

Notons $\theta_a = (\alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2)^\top$ et $\mathbf{X}_a = (x_1, y_1, x_2, y_2)^\top$ les angles et positions que l'on va estimer.

On sait que les droites de projection du point **X** sur les deux caméras sont sécantes (en **X**). On obtient alors facilement l'existence de λ et μ tels que[3] :

$$\tilde{\mathsf{C}}_1 + \lambda \mathsf{R}_1^{-1}\mathsf{K}^{-1}\mathbf{x}_1 = \tilde{\mathsf{C}}_2 + \mu \mathsf{R}_2^{-1}\mathsf{K}^{-1}\mathbf{x}_2 \ (= \tilde{\mathbf{X}})$$

Ainsi les trois vecteurs $\tilde{\mathsf{C}}_2 - \tilde{\mathsf{C}}_1$, $\mathsf{R}_2^{-1}\mathsf{K}^{-1}\mathbf{x}_2$ et $\mathsf{R}_1^{-1}\mathsf{K}^{-1}\mathbf{x}_1$ sont liés, ce qui se traduit par le fait que :

$$\det(\mathsf{R}_1^{-1}\mathsf{K}^{-1}\mathbf{x}_1 \mid \mathsf{R}_2^{-1}\mathsf{K}^{-1}\mathbf{x}_2 \mid \tilde{\mathsf{C}}_2 - \tilde{\mathsf{C}}_1) = 0 \tag{1}$$

Calculons les valeurs de $R_1^{-1}K^{-1}\mathbf{x}_1$ et $R_2^{-1}K^{-1}\mathbf{x}_2$:

Pour simplifier les expressions on notera c_θ pour $\cos(\theta)$ et s_θ pour $\sin(\theta)$. On a :

$$\begin{aligned}
\mathsf{R}_i^{-1}\mathsf{K}^{-1}\mathbf{x}_i &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{\alpha_i} & s_{\alpha_i} \\ 0 & -s_{\alpha_i} & c_{\alpha_i} \end{pmatrix} \begin{pmatrix} c_{\beta_i} & 0 & -s_{\beta_i} \\ 0 & 1 & 0 \\ s_{\beta_i} & 0 & c_{\beta_i} \end{pmatrix} \begin{pmatrix} c_{\gamma_i} & s_{\gamma_i} & 0 \\ -s_{\gamma_i} & c_{\gamma_i} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{\alpha_i} & s_{\alpha_i} \\ 0 & -s_{\alpha_i} & c_{\alpha_i} \end{pmatrix} \begin{pmatrix} c_{\beta_i} & 0 & -s_{\beta_i} \\ 0 & 1 & 0 \\ s_{\beta_i} & 0 & c_{\beta_i} \end{pmatrix} \begin{pmatrix} c_{\gamma_i} & s_{\gamma_i} & 0 \\ -s_{\gamma_i} & c_{\gamma_i} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i/f \\ y_i/f \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{\alpha_i} & s_{\alpha_i} \\ 0 & -s_{\alpha_i} & c_{\alpha_i} \end{pmatrix} \begin{pmatrix} c_{\beta_i} & 0 & -s_{\beta_i} \\ 0 & 1 & 0 \\ s_{\beta_i} & 0 & c_{\beta_i} \end{pmatrix} \begin{pmatrix} c_{\gamma_i}x_i/f + s_{\gamma_i}y_i/f \\ -s_{\gamma_i}x_i/f + c_{\gamma_i}y_i/f \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{\alpha_i} & s_{\alpha_i} \\ 0 & -s_{\alpha_i} & c_{\alpha_i} \end{pmatrix} \begin{pmatrix} c_{\beta_i}(c_{\gamma_i}x_i/f + s_{\gamma_i}y_i/f) - s_{\beta_i} \\ -s_{\gamma_i}x_i/f + c_{\gamma_i}y_i/f \\ s_{\beta_i}(c_{\gamma_i}x_i/f + s_{\gamma_i}y_i/f) + c_{\beta_i} \end{pmatrix} \\
&= \begin{pmatrix} c_{\beta_i}(c_{\gamma_i}x_i/f + s_{\gamma_i}y_i/f) - s_{\beta_i} \\ c_{\alpha_i}(-s_{\gamma_i}x_i/f + c_{\gamma_i}y_i/f) + s_{\alpha_i}[s_{\beta_i}(c_{\gamma_i}x_i/f + s_{\gamma_i}y_i/f) + c_{\beta_i}] \\ s_{\alpha_i}(s_{\gamma_i}x_i/f - c_{\gamma_i}y_i/f) + c_{\alpha_i}[s_{\beta_i}(c_{\gamma_i}x_i/f + s_{\gamma_i}y_i/f) + c_{\beta_i}] \end{pmatrix}
\end{aligned}$$

Nous verrons par la suite que l'on a surtout besoin de calculer les dérivées partielles du déterminant (1) par rapport aux variables α_i, β_i, γ_i, x_i et y_i.

Notons $\det(\mathsf{R}_1^{-1}\mathsf{K}^{-1}\mathbf{x}_1 \mid \mathsf{R}_2^{-1}\mathsf{K}^{-1}\mathbf{x}_2 \mid \tilde{\mathsf{C}_2} - \tilde{\mathsf{C}_1}) = \det \begin{pmatrix} k^1 & l^1 & m^1 \\ k^2 & l^2 & m^2 \\ k^3 & l^3 & m^3 \end{pmatrix}$.

[2] exceptionnellement on numérotera les caméras 1 et 2 et non 0 et 1, cela étant plus parlant dans une section théorique sans application directe en Python

[3] les $\sim$ indiquent qu'on utilise des coordonnées non homogènes

Considérons alors les mineurs d'ordre 2 des deux premières colonnes et calculons les dérivées partielles[4] :

- de $f^2 \det \begin{pmatrix} k^1 & l^1 \\ k^2 & l^2 \end{pmatrix}$:

  On rappelle que l'on a :

  - $[fk^1] = c_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) - fs_{\beta_1}$
  - $[fk^2] = c_{\alpha_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1) + s_{\alpha_1}[s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + fc_{\beta_1}]$
  - $[fl^1] = c_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) - fs_{\beta_2}$
  - $[fl^2] = c_{\alpha_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2) + s_{\alpha_2}[s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + fc_{\beta_2}]$

  On obtient :

Variable	Dérivée partielle
α_1	$-[fl^1](-s_{\alpha_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1) + c_{\alpha_1}[s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + fc_{\beta_1}])$
α_2	$[fk^1](-s_{\alpha_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2) + c_{\alpha_2}[s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + fc_{\beta_2}])$
β_1	$(-s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) - fc_{\beta_1})[fl^2] - [fl^1]s_{\alpha_1}[c_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) - fs_{\beta_1}]$
β_2	$[fk^1]s_{\alpha_2}[c_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) - fs_{\beta_2}] - (-s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) - fc_{\beta_2})[fk^2]$
γ_1	$c_{\beta_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1)[fl^2] - [fl^1](c_{\alpha_1}(-c_{\gamma_1}x_1 - s_{\gamma_1}y_1) + s_{\alpha_1}s_{\beta_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1))$
γ_2	$[fk^1](c_{\alpha_2}(-c_{\gamma_2}x_2 - s_{\gamma_2}y_2) + s_{\alpha_2}s_{\beta_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2)) - c_{\beta_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2)[fk^2]$
x_1	$c_{\beta_1}c_{\gamma_1}[fl^2] - [fl^1](-c_{\alpha_1}s_{\gamma_1} + s_{\alpha_1}s_{\beta_1}c_{\gamma_1})$
x_2	$[fk^1](-c_{\alpha_2}s_{\gamma_2} + s_{\alpha_2}s_{\beta_2}c_{\gamma_2}) - c_{\beta_2}c_{\gamma_2}[fk^2]$
y_1	$c_{\beta_1}s_{\gamma_1}[fl^2] - [fl^1](c_{\alpha_1}c_{\gamma_1} + s_{\alpha_1}s_{\beta_1}s_{\gamma_1})$
y_2	$[fk^1](c_{\alpha_2}c_{\gamma_2} + s_{\alpha_2}s_{\beta_2}s_{\gamma_2}) - c_{\beta_2}s_{\gamma_2}[fk^2]$

- de $f^2 \det \begin{pmatrix} k^1 & l^1 \\ k^3 & l^3 \end{pmatrix}$:

  On rappelle que l'on a :

  - $[fk^1] = c_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) - fs_{\beta_1}$
  - $[fk^3] = s_{\alpha_1}(s_{\gamma_1}x_1 - c_{\gamma_1}y_1) + c_{\alpha_1}[s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + fc_{\beta_1}]$
  - $[fl^1] = c_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) - fs_{\beta_2}$
  - $[fl^3] = s_{\alpha_2}(s_{\gamma_2}x_2 - c_{\gamma_2}y_2) + c_{\alpha_2}[s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + fc_{\beta_2}]$

  On obtient :

Variable	Dérivée partielle
α_1	$-[fl^1](c_{\alpha_1}(s_{\gamma_1}x_1 - c_{\gamma_1}y_1) - s_{\alpha_1}[s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + fc_{\beta_1}])$
α_2	$[fk^1](c_{\alpha_2}(s_{\gamma_2}x_2 - c_{\gamma_2}y_2) - s_{\alpha_2}[s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + fc_{\beta_2}])$
β_1	$(-s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) - fc_{\beta_1})[fl^3] - [fl^1]c_{\alpha_1}[c_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) - fs_{\beta_1}]$
β_2	$[fk^1]c_{\alpha_2}[c_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) - fs_{\beta_2}] - (-s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) - fc_{\beta_2})[fk^3]$
γ_1	$c_{\beta_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1)[fl^3] - [fl^1](s_{\alpha_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + c_{\alpha_1}[s_{\beta_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1))$
γ_2	$[fk^1](s_{\alpha_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + c_{\alpha_2}[s_{\beta_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2)]) - c_{\beta_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2)[fk^3]$
x_1	$c_{\beta_1}c_{\gamma_1}[fl^3] - [fl^1](s_{\alpha_1}s_{\gamma_1} + c_{\alpha_1}s_{\beta_1}c_{\gamma_1})$
x_2	$[fk^1](s_{\alpha_2}s_{\gamma_2} + c_{\alpha_2}s_{\beta_2}c_{\gamma_2}) - c_{\beta_2}c_{\gamma_2}[fk^3]$
y_1	$c_{\beta_1}s_{\gamma_1}[fl^3] - [fl^1](-s_{\alpha_1}c_{\gamma_1} + c_{\alpha_1}s_{\beta_1}s_{\gamma_1})$
y_2	$[fk^1](-s_{\alpha_2}c_{\gamma_2} + c_{\alpha_2}s_{\beta_2}s_{\gamma_2}) - c_{\beta_2}s_{\gamma_2}[fk^3]$

[4] on multiplie par f^2 pour simplifier les formules, cela étant sans importance

- de $f^2 \det \begin{pmatrix} k^2 & l^2 \\ k^3 & l^3 \end{pmatrix}$:

  On rappelle que l'on a :

  - $[fk^2] = c_{\alpha_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1) + s_{\alpha_1}[s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + fc_{\beta_1}]$
  - $[fk^3] = s_{\alpha_1}(s_{\gamma_1}x_1 - c_{\gamma_1}y_1) + c_{\alpha_1}[s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + fc_{\beta_1}]$
  - $[fl^2] = c_{\alpha_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2) + s_{\alpha_2}[s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + fc_{\beta_2}]$
  - $[fl^3] = s_{\alpha_2}(s_{\gamma_2}x_2 - c_{\gamma_2}y_2) + c_{\alpha_2}[s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + fc_{\beta_2}]$

  On obtient :

Variable	Dérivée partielle
α_1	$(-s_{\alpha_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1) + c_{\alpha_1}[s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + fc_{\beta_1}])[fl^3]$ $-[fl^2](c_{\alpha_1}(s_{\gamma_1}x_1 - c_{\gamma_1}y_1) - s_{\alpha_1}[s_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1))$
α_2	$[fk^2](c_{\alpha_2}(s_{\gamma_2}x_2 - c_{\gamma_2}y_2) - s_{\alpha_2}[s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + fc_{\beta_2}])$ $-(-s_{\alpha_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2) + c_{\alpha_2}[s_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + fc_{\beta_2}])[fk^3]$
β_1	$s_{\alpha_1}[c_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) - fs_{\beta_1}][fl^3]$ $-[fl^2]c_{\alpha_1}[c_{\beta_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) - fs_{\beta_1}]$
β_2	$[fk^2]c_{\alpha_2}[c_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) - fs_{\beta_2}]$ $-(s_{\alpha_2}[c_{\beta_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) - fs_{\beta_2}])[fk^3]$
γ_1	$(c_{\alpha_1}(-c_{\gamma_1}x_1 - s_{\gamma_1}y_1) + s_{\alpha_1}s_{\beta_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1))[fl^3]$ $-[fl^2](s_{\alpha_1}(c_{\gamma_1}x_1 + s_{\gamma_1}y_1) + c_{\alpha_1}s_{\beta_1}(-s_{\gamma_1}x_1 + c_{\gamma_1}y_1))$
γ_2	$[fk^2](s_{\alpha_2}(c_{\gamma_2}x_2 + s_{\gamma_2}y_2) + c_{\alpha_2}s_{\beta_2}(s_{\gamma_2}x_2 - c_{\gamma_2}y_2))$ $-(c_{\alpha_2}(-c_{\gamma_2}x_2 - s_{\gamma_2}y_2) + s_{\alpha_2}s_{\beta_2}(-s_{\gamma_2}x_2 + c_{\gamma_2}y_2))[fk^3]$
x_1	$(-c_{\alpha_1}s_{\gamma_1} + s_{\alpha_1}s_{\beta_1}c_{\gamma_1})[fl^3] - [fl^2](s_{\alpha_1}s_{\gamma_1} + c_{\alpha_1}s_{\beta_1}c_{\gamma_1})$
x_2	$[fk^2](s_{\alpha_2}s_{\gamma_2} + c_{\alpha_2}s_{\beta_2}c_{\gamma_2}) - (-c_{\alpha_2}s_{\gamma_2} + s_{\alpha_2}s_{\beta_2}c_{\gamma_2})[fk^3]$
y_1	$(c_{\alpha_1}c_{\gamma_1} + s_{\alpha_1}s_{\gamma_1})[fl^3] - [fl^2](-s_{\alpha_1}c_{\gamma_1} + c_{\alpha_1}s_{\beta_1}s_{\gamma_1})$
y_2	$[fk^2](-s_{\alpha_2}c_{\gamma_2} + c_{\alpha_2}s_{\beta_2}s_{\gamma_2}) - (c_{\alpha_2}c_{\gamma_2}y_2 + s_{\alpha_2}s_{\beta_2}s_{\gamma_2})[fk^3]$

On utilise alors la formule :

$$\det(\mathsf{R}_1^{-1}\mathsf{K}^{-1}\mathbf{x}_1 \mid \mathsf{R}_2^{-1}\mathsf{K}^{-1}\mathbf{x}_2 \mid \tilde{\mathsf{C}}_2 - \tilde{\mathsf{C}}_1) = m_1(k_2l_3 - k_3l_2) - m_2(k_1l_3 - k_3l_1) + m_3(k_1l_2 - k_2l_1)$$

où $m_1 = x_{C_2} - x_{C_1}$, $m_2 = y_{C_2} - y_{C_1}$ et $m_3 = z_{C_2} - z_{C_1}$

On obtient alors les dérivées partielles par rapport à chacune des variables de notre déterminant en prenant les combinaisons linéaires des trois tableaux précédents.

2.2 Généralisation

Lorsque l'on a plusieurs caméras et plusieurs points, on doit chercher $(\theta_a, \mathbf{X}_a)$ qui annulent tous les déterminants entre deux matrices et un point.

Posons alors $F : \begin{array}{ccc} \mathbb{R}^{3K} \times \mathbb{R}^{2NK} & \longrightarrow & \mathbb{R}^{\binom{K}{2}N} \\ (\theta, \mathbf{X}) & \longmapsto & (\det^j_{i_1,i_2})_{1 \leqslant i_1 < i_2 \leqslant K,\ 0 \leqslant j \leqslant N-1} \end{array}$ où $\det^j_{i_1,i_2}$ est le déterminant de la forme (1) associé aux caméras i_1, i_2 et au point j.

Notre problème consiste donc à chercher $(\theta_a, \mathbf{X}_a)$ qui annulent F au voisinage de $(\theta_0, \mathbf{X}_0)$.

2.2.1 Stratégie

On applique une formule de TAYLOR à l'ordre 1 :

$$F(\theta, \mathbf{X}) = F(\theta_0, \mathbf{X}_0) + dF_\theta(\theta_0, \mathbf{X}_0)d\theta + dF_{\mathbf{X}}(\theta_0, \mathbf{X}_0)d\mathbf{X} + o((d\theta, d\mathbf{X}))$$

où $d\theta = \theta - \theta_0$ et $d\mathbf{X} = \mathbf{X} - \mathbf{X}_0$.

On résout alors $F(\theta, \mathbf{X}) = 0$ au premier ordre, c'est-à-dire on cherche $(d\theta, d\mathbf{X})$ tels que :

$$\left(\frac{\partial F}{\partial \theta}(\theta_0, \mathbf{X}_0), \frac{\partial F}{\partial \mathbf{X}}(\theta_0, \mathbf{X}_0)\right) \begin{pmatrix} d\theta \\ d\mathbf{X} \end{pmatrix} = -F(\theta_0, \mathbf{X}_0)$$

Par la suite, on posera $A = \dfrac{\partial F}{\partial \theta}(\theta_0, \mathbf{X}_0)$ et $B = \dfrac{\partial F}{\partial \mathbf{X}}(\theta_0, \mathbf{X}_0)$.

Notre problème consiste donc à inverser la matrice $M := (A, B)$. Cependant, comme on va avoir beaucoup de points de correspondance, on aura beaucoup plus d'équations que d'inconnues (le système est sur-déterminé), et donc notre matrice ne sera pas inversible. Cependant, on espère que les équations à première vue incompatibles seront globalement redondantes et nous permettrons ainsi de gagner en précision. On va donc calculer la pseudo-inverse de M que l'on notera M^+.

La matrice $-M^+F(\theta_0, \mathbf{X}_0)$ fournit alors la meilleure approximation de $\begin{pmatrix} d\theta \\ d\mathbf{X} \end{pmatrix}$ au sens des moindres carrés.

On peut montrer que dans notre cas **[M est une matrice réelle et $M^\top M$ est inversible (?)]** :

$$M^+ = (M^\top M)^{-1} M^\top$$

Algorithmiquement, la pseudo-inverse s'obtient à partir de la décomposition en valeurs singulières de M. On décompose $M = U\Sigma V^\top$ puis on calcule $M^+ = V\Sigma^+ U^\top$, où Σ^+ est l'inverse de la matrice diagonale Σ et s'obtient donc aisément.

Remarque Des bibliothèques Python permettrons de calculer les pseudo-inverses aisément.

2.2.2 Condition nécessaire sur le nombre de caméras

Pour calculer la pseudo inverse de $M = (A, B)$, on doit avoir plus d'équations que d'inconnues, c'est-à-dire que M doit avoir plus de lignes que de colonnes.

On rappelle que M possède $\binom{K}{2}N$ lignes et $3K + 2NK$ colonnes.

On résout donc $\binom{K}{2}n \geqslant (3 + 2N)K$, et on obtient :

$$\boxed{N(K-5) \geqslant 6}$$

Il faut donc au moins 6 caméras pour espérer calculer la pseudo inverse. [faire une phrase d'explications]

3 Étude des matrices jacobiennes partielles A et B

3.1 Étude de la matrice A

A est la matrice jacobienne $D_\theta F \in \mathcal{M}_{\binom{K}{2}N,3K}$. On s'intéresse donc aux dérivées de F par rapport à tous les angles.

On remarque que les $\det^j_{i_1,i_2}$ ne dépendent que des angles α_{i_1}, β_{i_1}, γ_{i_1}, α_{i_2}, β_{i_2} et γ_{i_2}, et donc leurs dérivées par rapport à tous les angles associés aux caméras i telles que $i \neq i_1$ et $i \neq i_2$. Ainsi, la matrice A possède de nombreux 0. Essayons de comprendre leur emplacement.

Pour commencer, choisissons une convention pour l'ordre des composantes de F (lignes de A) et des angles de θ (colonnes de A).

- Concernant les lignes, on associera aux lignes l comprises entre $\binom{K}{2}j \leqslant l < \binom{K}{2}(j+1)$ les déterminants associés au j-ième point de correspondance ($0 \leqslant j \leqslant N-1$), c'est-à-dire de la forme $\det^j_{i_1,i_2}$. Reste à choisir comment organiser au sein de ces groupes de lignes l'ordre des indices (i_1, i_2). On écrit les lignes dans l'ordre $(0,1), (0,2), \dots (0,K-1), (1,2), (1,3), \dots (1,K-1), \dots\dots (K-2,K-1)$.
- Concernant les colonnes, on dérivera pour A les composantes par rapport aux variables suivant l'ordre $\underbrace{(\alpha_0, \beta_0, \gamma_0, \alpha_1, \beta_1, \gamma_1, \dots, \alpha_{K-1}, \beta_{K-1}, \gamma_{K-1})}_{\theta^\top}$.

Exemple dans le cas $K = 4$ La matrice A est de la forme :

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_j \\ \vdots \\ A_N \end{pmatrix} \text{ où } A_j = \begin{array}{c} \begin{array}{cccccccccccc} l & \alpha_0 & \beta_0 & \gamma_0 & \alpha_1 & \beta_1 & \gamma_1 & \alpha_2 & \beta_2 & \gamma_2 & \alpha_3 & \beta_3 & \gamma_3 \end{array} \\ \begin{pmatrix} 6j & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 \\ 6j+1 & & & & 0 & 0 & 0 & & & & 0 & 0 & 0 \\ 6j+2 & & & & 0 & 0 & 0 & 0 & 0 & 0 & & & \\ 6j+3 & 0 & 0 & 0 & & & & & & & 0 & 0 & 0 \\ 6j+4 & 0 & 0 & 0 & & & & 0 & 0 & 0 & & & \\ 6j+5 & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & \end{pmatrix} \end{array} \begin{array}{c} (i_1,i_2) \\ (0,1) \\ (0,2) \\ (0,3) \\ (1,2) \\ (1,3) \\ (2,3) \end{array}$$

Dans le cas général, A est composé des matrices A_j de taille $\binom{K}{2}$x$3K$.

Ensuite, les coefficients non nuls de A se calculent en utilisant les formules de la section précédente.

3.2 Étude de la matrice B

Commençons, comme pour A, par choisir une convention pour l'ordre des composantes de F (lignes de B) et des perturbations de δ (colonnes de B) :

- Concernant les lignes, on garde la même convention que pour A,
- Concernant les colonnes, on dérivera les composantes par rapport aux variables suivant l'ordre $\underbrace{\left(x_0^0, y_0^0, \dots, x_i^0, y_i^0, \dots, x_{K-1}^0, y_{K-1}^0, \dots\dots, x_0^j, y_0^j, \dots, x_{K-1}^j, y_{K-1}^j, \dots\dots, x_0^{N-1}, y_0^{N-1}, \dots, x_{K-1}^{N-1}, y_{K-1}^{N-1}\right)}_{\delta^\top}$.

Avec cette convention, on peut a encore un bon aperçu de la forme de B :

Exemple dans le cas $K = 4$ La matrice B est de la forme :

$$B = \text{diag}(B_1, \dots, B_j, \dots, B_N) \text{ où } B_j = \begin{array}{c} l \\ 6j \\ 6j+1 \\ 6j+2 \\ 6j+3 \\ 6j+4 \\ 6j+5 \end{array} \begin{array}{c} \begin{array}{cccccccc} x_0^j & y_0^j & x_1^j & y_1^j & x_2^j & y_2^j & x_3^j & y_3^j \end{array} \\ \begin{pmatrix} & & & & 0 & 0 & 0 & 0 \\ & & 0 & 0 & & & 0 & 0 \\ & & 0 & 0 & 0 & 0 & & \\ 0 & 0 & & & & & 0 & 0 \\ 0 & 0 & & & 0 & 0 & & \\ 0 & 0 & 0 & 0 & & & & \end{pmatrix} \end{array} \begin{array}{c} (i_1, i_2) \\ (1,2) \\ (1,3) \\ (1,4) \\ (2,3) \\ (2,4) \\ (3,4) \end{array}$$

Dans le cas général, B est donc une matrice diagonale par blocs et on a $B_j \in \mathcal{M}_{\binom{K}{2} \times 2N}$.

De même, les coefficients non nuls de B se calculent en utilisant les formules de la section précédente.

Par la suite, et comme nous allons le voir dans nos simulations, nous aurons besoin d'étudier notamment les conditionnements des matrices A et B.

4 Premières simulations

Avant d'appliquer notre méthode sur des images satellites réelles, nous devons la tester avec des simulations. Nous avons utilisé pour cela Python.

On trouvera en annexe les codes relatifs à cette partie.

4.1 Le cas satellitaire

4.1.1 Génération de données

Dans la réalité, un satellite situé à environ 800 km de la Terre capture des images terrestres d'environ 20 km par 20 km.

Afin de simuler au mieux des données réelles, on souhaite donc modéliser des scènes vérifiant les critères suivants :

- on veut que les caméras visent plus ou moins un même point au centre de la zone à modéliser, que l'on appelera point de visée,
- les points de correspondance identifiés par les alogrithmes tels que la méthode SIFT doivent être situés dans la zone de 20 km par 20 km. Cependant, il ne faut pas négliger le relief terrestre : les points ne sont pas sur un plan, on leur donne donc une liberté supplémentaire sur leur altitude qui doit pouvoir varier de quelques kilomètres.

On génère ainsi aléatoirement N points de correspondance et K caméras correspondants à une telle situation.

L'étape la plus compliquée consisteà choisir les angles associés aux caméras pour qu'elles pointent le point de visée, que l'on définit arbitrairement comme le point $O = (0, 0, 0)$ pour simplifier les calculs. Il est alors facile de voir que, pour toute caméra i, le vecteur normalisé $-\frac{\vec{OC}}{\|\vec{OC}\|}$ doit être l'image de Z_T par la matrice de rotation R_k associée à la caméra. De plus, on remarque que l'on peut fixer l'angle β à 0 sans pour autant empêcher que certains vecteurs soient l'image de Z_T par R_k. Avec ce choix, on détermine alors aisément et de manière unique α et γ.

On obtient avec notre code des situations comme celle-ci :

Remarque Évidemment l'échelle de variation des points de correspondance est très faible par rapport à l'altitude des satellites. Les points semblent donc être sur un même plan. C'est une spécificité du cas satellitaire sur laquelle nous reviendrons plus tard.

4.1.2 Perturbation des données

A partir des points et des caméras créés, on peut, en utilisant les matrices de projection des caméras, obtenir les coordonnées réelles des images des points sur nos caméras. On connaît donc les valeurs réelles des angles θ_* et des images $\mathbf{X}_*$ (cf. partie 1.3).

On perturbe alors ces points et ces angles en ajoutant un bruit gaussien de paramètres $\sigma_{\mathbf{X}}$ et σ_θ. Typiquement, on a $\sigma_{\mathbf{X}} \simeq 10^{-2}$ px et $\sigma_\theta \simeq 10^{-5}$ rad.

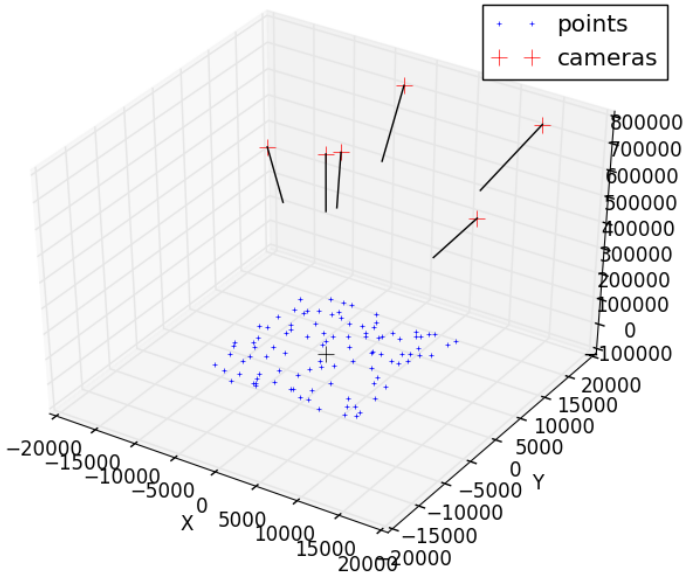

Figure 2: Une situation avec 6 caméras et 100 points
(les traits noirs sont les directions de visée des caméras)

On obtient alors θ_0 et $\mathbf{X}_0$.

Remarque On rappelle que dans la réalité, on n'a pas accès à θ_* et $\mathbf{X}_*$, on nous fournit directement θ_0 et $\mathbf{X}_0$. Ainsi, dans la suite du code, on ne pourra bien évidemment pas utiliser θ_* et $\mathbf{X}_*$ pour nos calculs. On ne s'en servira que pour vérifier si nos estimations finales θ_a et $\mathbf{X}_a$ sont plus proches ou non des données réelles que ne le sont θ_0 et $\mathbf{X}_0$.

4.1.3 Estimation des paramètres réels

Ensuite, on utilise les données perturbées θ_0 et $\mathbf{X}_0$ à notre disposition pour calculer les matrices A et B, et donc la matrice M.

Sa pseudo-inverse permet alors de calculer nos estimations θ_a et $\mathbf{X}_a$.

Reste à vérifier que l'on obtient des résultat cohérents, c'est-à-dire que θ_a et $\mathbf{X}_a$ soient plus proches de θ_* et $\mathbf{X}_*$ que θ_0 et $\mathbf{X}_0$ et que $\|F(\theta_a, \mathbf{X}_a)\| \ll \|F(\theta_0, \mathbf{X}_0)\|$.

4.1.4 Premiers résultats

En réalisant plusieurs expériences, et en faisant notamment varier le nombre de caméras, de points, les paramètres $\sigma_{\mathbf{X}}$ et σ_θ, on s'aperçoit :

- qu'en général, on observe bien que $\|F(\theta_a, \mathbf{X}_a)\| \ll \|F(\theta_0, \mathbf{X}_0)\|$, mais cela n'est pas automatique,
- que $\|\theta_a - \theta_*\| > \|\theta_0 - \theta_*\|$ et $\|\mathbf{X}_a - \mathbf{X}_*\| > \|\mathbf{X}_0 - \mathbf{X}_*\|$ la plupart du temps.

On remarque donc qu'on a un problème puisque l'on n'arrive pas à se rapprocher des paramètres réels. Dans certain cas, on n'arrive quand même à diminuer significativement les valeurs de F, ce qui est mauvais signe puisque cela pourrait signifier que F prend des valeurs très faibles sur toute une famille de solutions proche de notre solution réelle, et donc qu'il est difficilement possible de retrouver la valeur attendue.

4.1.5 Étude plus approfondie : le conditionnement des matrices

Afin de déterminer la nature du problème, on le teste dans des situations moins compliquées : avec moins de caméras, moins de points ... **[à compléter]**

Une expérience importante consiste à voir ce qui se passe lorsque l'on essaye d'estimer uniquement les angles ou uniquement les coordonnées des images.

Tout d'abord, si l'**on fixe les coordonnées des images** et qu'**on essaye d'estimer uniquement les angles**, la matrice associée au problème (c'est-à-dire dont on doit calculer la pseudo-inverse) est la matrice A.

- Si l'on suppose que les coordonnées ne sont pas perturbées et donc que $\mathbf{X}_0 = \mathbf{X}_*$, on remarque que l'on arrive toujours à retrouver les angles réels des caméras avec une très grande précision.

  Pour différentes valeurs de N, on réalise une dizaine de simulations (avec des coordonnées différentes à chaque fois) puis on calcule les normes moyennes des vecteurs $\theta_a - \theta_*$ puis $F(\theta_a, \mathbf{X}_0 = \mathbf{X}_*)$ (en rouge ci-dessous) et $\theta_0 - \theta_*$ puis $F(\theta_0, \mathbf{X}_0 = \mathbf{X}_*)$ (en bleu).

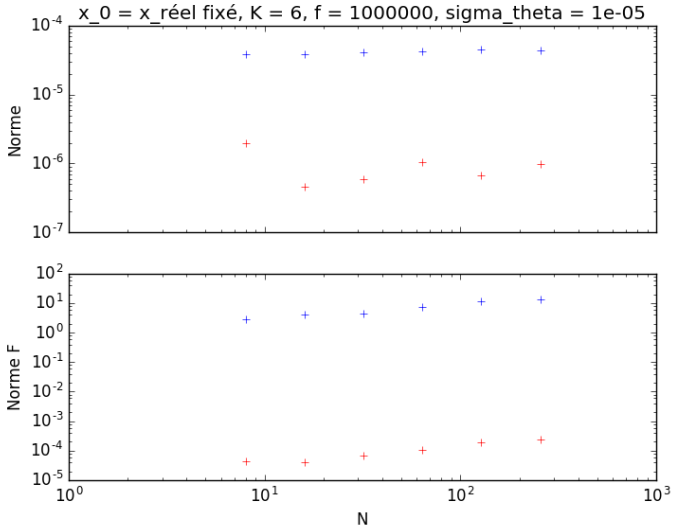

Figure 3: Évolution des différentes normes associées au problème en fonction du nombre de points

  On voit que notre estimation finale est beaucoup plus proche de la valeur réelle et que la norme de F a été fortement diminuée.

- On peut aussi supposer qu'on a eu une perturbation des coordonnées des images mais qu'on ne cherche pas à la corriger. Dans ce cas, on a donc $\mathbf{X}_0 \neq \mathbf{X}_*$. On obtient alors des résultats du type :

  Cette fois-ci, on n'arrive pas à estimer correctement les angles, et même on n'arrive pas à diminuer la norme de F.

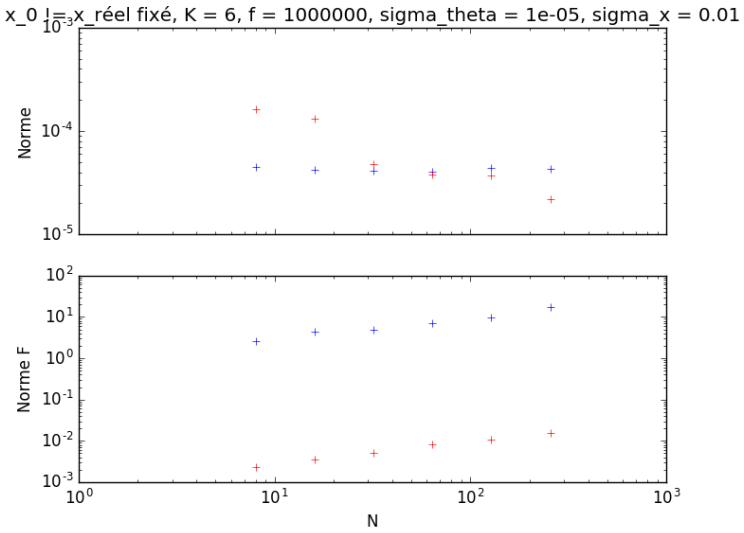

Figure 4: Évolution des différentes normes associées au problème en fonction du nombre de points

Ensuite, on peut faire l'expérience inverse : on fixe les angles des caméras et on essaye d'estimer uniquement les coordonnées des images. La matrice associée au problème est alors la matrice B.

Si l'on suppose $\theta_0 = \theta_*$ (on n'a pas d'erreur sur nos angles), on obtient par exemple :

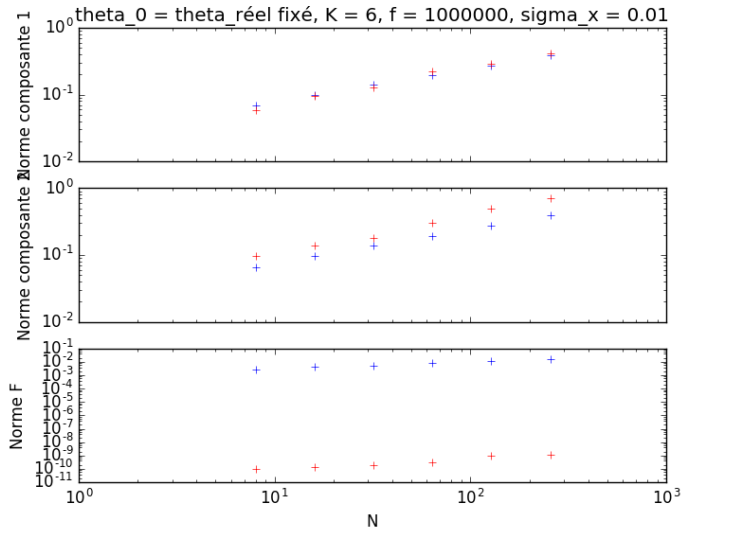

Figure 5: Évolution des différentes normes associées au problème en fonction du nombre de points

Ici, on arrive bien à minimiser F mais on n'arrive pas à se rapprocher des coordonnées réelles.

Comment interpréter ces résultats ?

→ On peut s'attendre à un **problème de conditionnement** des différentes matrices.

Regardons alors les conditionnements des matrices A, B et M, pour une dizaine de matrices aléatoires pour chaque valeur de N testée :

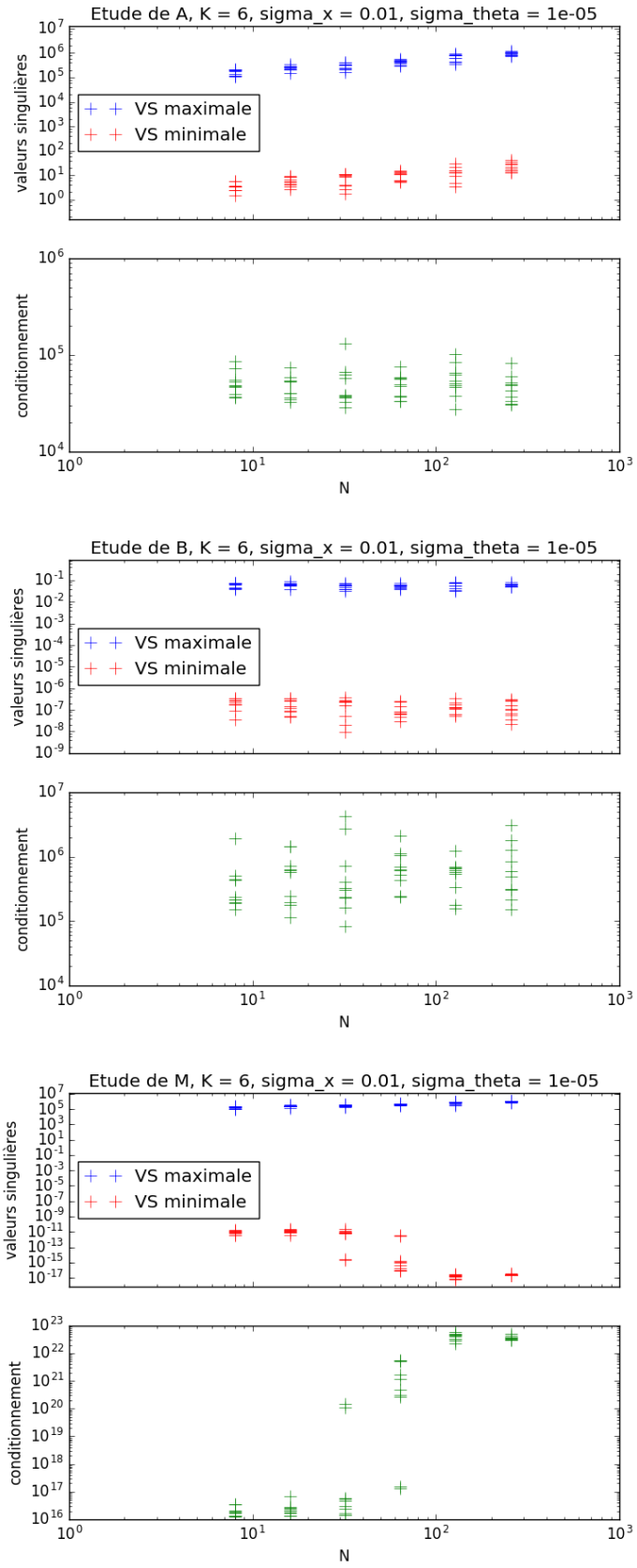

Figure 6: Étude des conditionnements des matrices $A^\top A$, $B^\top B$ et $M^\top M$

On observe de très mauvais conditionnements pour le problème général associé à la matrice M (où l'on doit réestimer tous les paramètres). Les deux autres conditionnements pour les problèmes avec paramètres fixés

sont moins monstrueux mais restent très importants.

4.2 Un cas terrestre

Essayons de voir ce que donne notre méthode si l'on change les ordres de grandeurs, en essayant par exemple de reconstituer une scène avec des images prises par des drones.

4.2.1 Génération de données

On procède un peu de la même manière pour générer nos données. Évidemment, on change tous les ordres de grandeurs : la scène s'étend sur une zone d'environ 100 m par 100 m, sur une hauteur d'une quarantaine de mètres environ.

On obtient avec notre code des situations comme celle-ci :

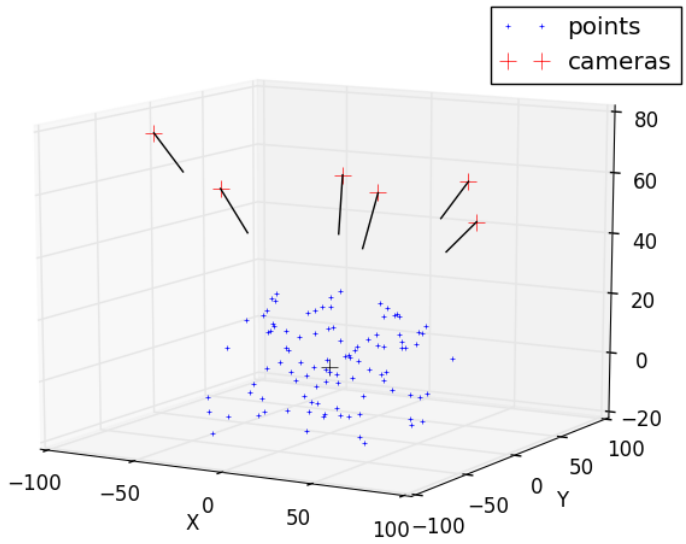

Figure 7: Une situation avec 6 caméras et 100 points
(les traits noirs sont les directions de visée des caméras)

Cette fois-ci, les points de correspondance se situent dans un pavé et ne semblent plus être sur un même plan. Cela laisse peut-être présager un meilleur conditionnement des matrices.

4.2.2 Perturbation des données

On perturbe de même nos données en ajoutant un bruit gaussien de paramètres $\sigma_{\mathbf{X}}$ et σ_θ.

Les ordres de grandeurs des perturbations sont ici différents, on prend typiquement $\sigma_{\mathbf{X}} \simeq 10^{-2}$ px et $\sigma_\theta \simeq 10^{-2}$ rad.

4.2.3 Estimation des paramètres réels

Là encore, on utilise les données perturbées θ_0 et $\mathbf{X}_0$ à notre disposition pour calculer les matrices A et B, et donc la matrice M.

Sa pseudo-inverse permet alors de calculer nos estimations θ_a et $\mathbf{X}_a$.

4.2.4 Premiers résultats

En réalisant plusieurs expériences, on observe globalement les mêmes résultats que pour le cas satellitaire, à savoir qu'en faisant notamment varier le nombre de caméras, de points, les paramètres $\sigma_{\mathbf{X}}$ et σ_θ, on s'aperçoit :

- qu'en général, on observe bien que $\|F(\theta_a, \mathbf{X}_a)\| \ll \|F(\theta_0, \mathbf{X}_0)\|$, mais cela n'est pas automatique,
- que $\|\theta_a - \theta_*\| > \|\theta_0 - \theta_*\|$ et $\|\mathbf{X}_a - \mathbf{X}_*\| > \|\mathbf{X}_0 - \mathbf{X}_*\|$ la plupart du temps.

4.2.5 Étude plus approfondie : le conditionnement des matrices

Tout d'abord, si l'**on fixe les coordonnées des images** et qu'**on essaye d'estimer uniquement les angles**, la matrice associée au problème (c'est-à-dire dont on doit calculer la pseudo-inverse) est la matrice A.

- Si l'on suppose que les coordonnées ne sont pas perturbées et donc que $\mathbf{X}_0 = \mathbf{X}_*$, on remarque que l'on arrive toujours à retrouver les angles réels des caméras avec une très grande précision.

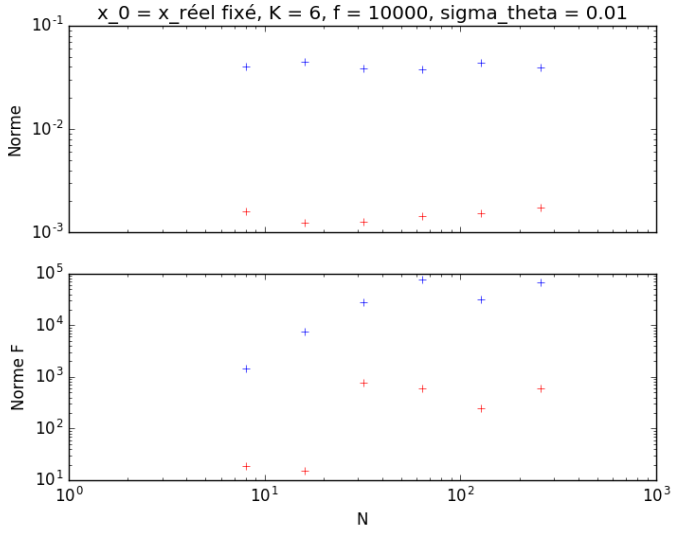

Figure 8: Évolution des différentes normes associées au problème en fonction du nombre de points

On voit que notre estimation finale est beaucoup plus proche de la valeur réelle et que la norme de F a été fortement diminuée.

- On peut aussi supposer qu'on a eu une perturbation des coordonnées des images mais qu'on ne cherche pas à la corriger. Dans ce cas, on a donc $\mathbf{X}_0 \neq \mathbf{X}_*$. On obtient alors des résultats du type :

  On arrive donc une nouvelle fois à retrouver les paramètres angulaires réels avec une bonne précision. **C'est mieux que dans le cas satellitaire, puisque cette expérience ne fonctionnait qu'à partir d'un nombre important de points !**

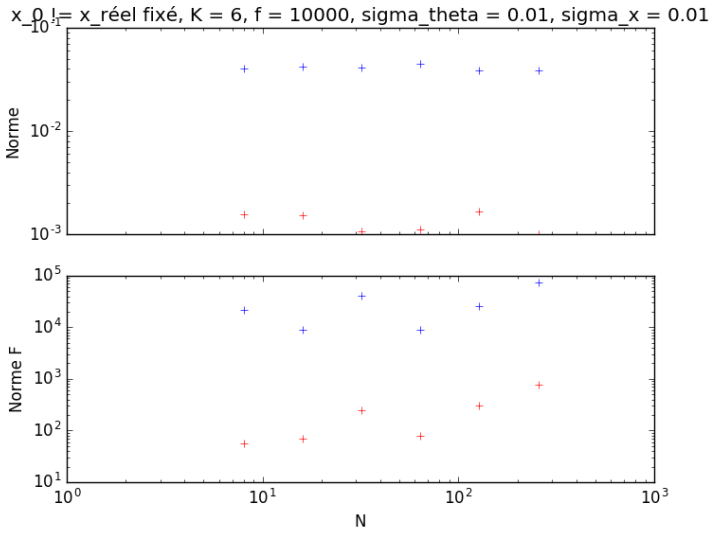

Figure 9: Évolution des différentes normes associées au problème en fonction du nombre de points

Ensuite, on peut faire l'expérience inverse : on fixe les angles des caméras et on essaye d'estimer uniquement les coordonnées des images. La matrice associée au problème est alors la matrice B.

Si l'on suppose $\theta_0 = \theta_*$ (on n'a pas d'erreur sur nos angles), on obtient par exemple :

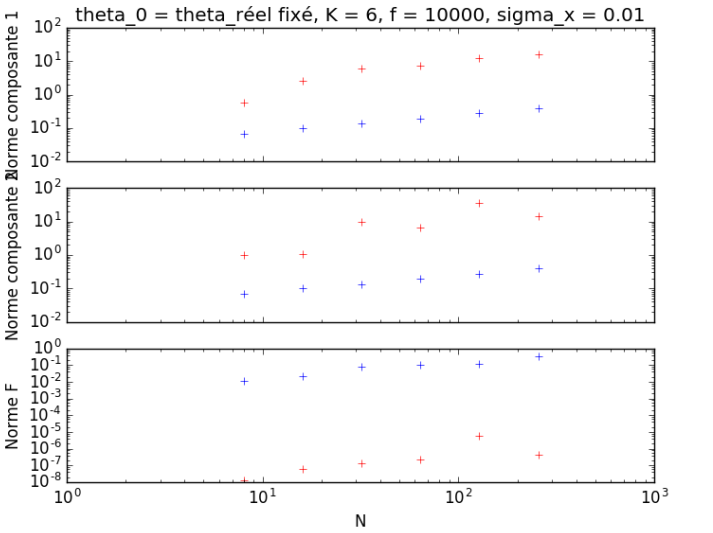

Figure 10: Évolution des différentes normes associées au problème en fonction du nombre de points

Comme dans le cas satellitaire, on arrive bien à minimiser F mais on n'arrive pas à se rapprocher des coordonnées réelles.

Regardons alors les conditionnements des matrices A, B et M, pour une dizaine de matrices aléatoires pour chaque valeur de N testée :

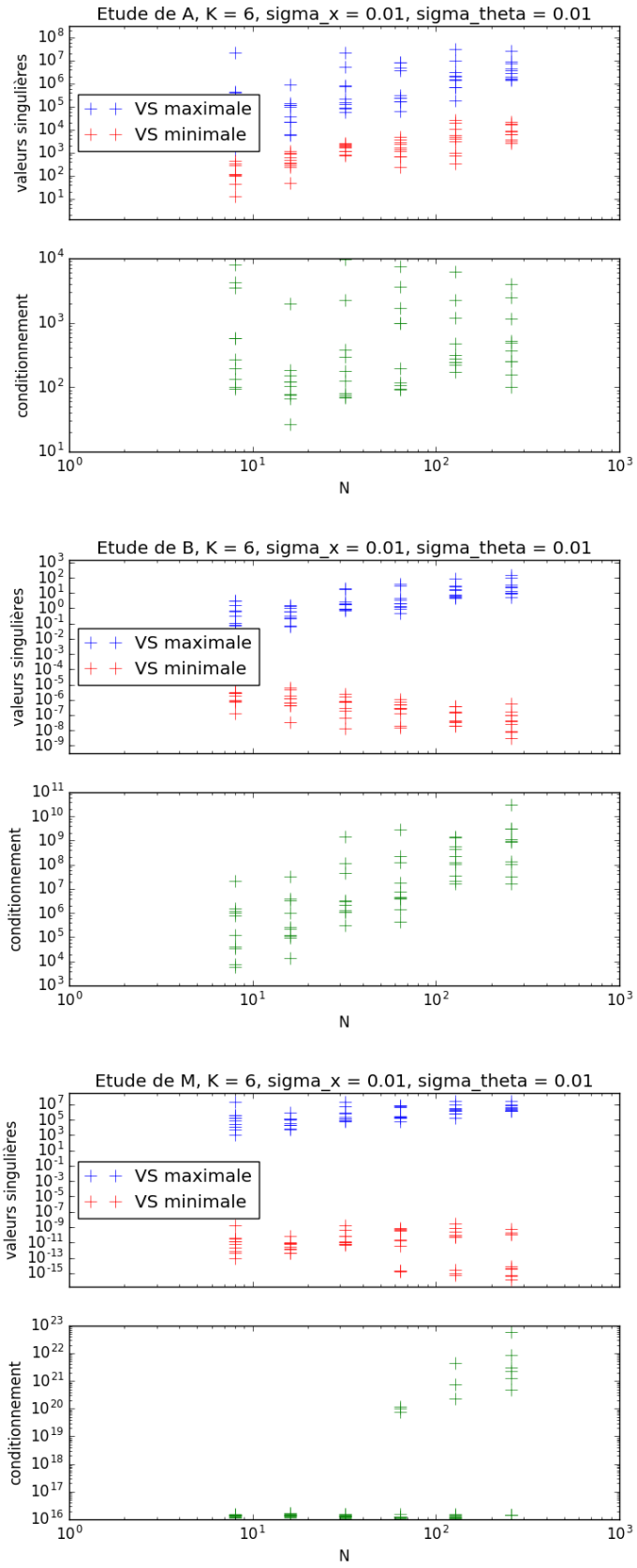

Figure 11: Étude des conditionnements des matrices $A^\top A$, $B^\top B$ et $M^\top M$

On observe encore de très mauvais conditionnements pour le problème général associé à la matrice M. Le conditionnement de la matrice B (estimation des coordonnées des images) est lui aussi assez mauvais. Pour la

matrice A, le conditionnement est assez aléatoire mais pas trop important.

5 Améliorations de la méthode

5.1 Ajout d'un terme forçant

5.1.1 Principe

Par les expériences précédentes, on constate que notre système linéaire possède plus d'une solution. On espérait que la résolution, au sens des moindres carrés, du système

$$M \begin{pmatrix} d\theta \\ d\mathbf{X} \end{pmatrix} = -F(\theta_0, \mathbf{X}_0)$$

nous conduirait à l'obtention d'une unique solution, à savoir la solution $(\theta_*, \mathbf{X}_*)$ mais ce n'est clairement pas le cas puisque le simulateur nous renvoie des valeurs $(\theta_a, \mathbf{X}_a)$ qui diminuent la norme de F mais qui sont loin à la fois de la solution exacte et de la solution initiale.

Une idée consiste donc à imposer une contrainte supplémentaire au problème : on cherche des solutions $(d\theta, d\mathbf{X})$ qui minimisent la quantité :

$$\left\| M \begin{pmatrix} d\theta \\ d\mathbf{X} \end{pmatrix} + F(\theta_0, \mathbf{X}_0) \right\|^2 + \lambda^2 \frac{1}{\sigma^2} \left\| \begin{pmatrix} d\theta \\ d\mathbf{X} \end{pmatrix} \right\|^2 \tag{2}$$

où λ est un préfacteur à déterminer afin d'obtenir des résultats optimaux.

Cette nouvelle contrainte permet de mieux contrôler la distance entre $(\theta_0, \mathbf{X}_0)$ et $(\theta_a, \mathbf{X}_a)$, et donc on l'espère d'obtenir une approximation finale meilleure que sur nos premières simulations.

Remarque En pratique, on n'a pas $\sigma_{\mathbf{X}} = \sigma_\theta$, et on n'est pas obligé d'imposer $\lambda_{\mathbf{X}} = \lambda_\theta$. On essayera donc plutôt de minimiser la quantité :

$$\left\| M \begin{pmatrix} d\theta \\ d\mathbf{X} \end{pmatrix} + F(\theta_0, \mathbf{X}_0) \right\|^2 + \frac{\lambda_\theta^2}{\sigma_\theta^2} \|d\theta\|^2 + \frac{\lambda_{\mathbf{X}}^2}{\sigma_{\mathbf{X}}^2} \|d\mathbf{X}\|^2$$

5.1.2 Implémentation

Pour mettre en place cette méthode, on rajoute à notre système linéaire les équations : $\frac{\lambda_\theta}{\sigma_\theta} d\theta = 0$ et $\frac{\lambda_{\mathbf{X}}}{\sigma_{\mathbf{X}}} d\mathbf{X} = 0$.

Cela revient à rajouter le bloc $\mathrm{diag}(\sigma_\theta, \dots, \sigma_\theta, \sigma_{\mathbf{X}}, \dots, \sigma_{\mathbf{X}}) \in \mathcal{M}_{3K+2KN}$ à la matrice M. On obtient ainsi une nouvelle matrice M' qui possède plus de lignes (plus d'équations) et on résout le système par la méthode habituelle, la pseudo-inverse de M' permettant alors de minimiser la quantité (2).

Remarque Algorithmiquement, cela rallonge considérablement le temps de calcul.

5.1.3 Résultats

Dans la suite, on utilisera les indices suivants :

- 0 sera associé aux données initialement perturbées,

- ”reel” sera associé aux données exactes,
- a_1 sera associé à la solution fournie par l'algorithme sans terme forçant,
- a_2 sera associé à la solution fournie par l'algorithme avec terme forçant.

Commençons par vérifier quelle méthode minimise au mieux F :

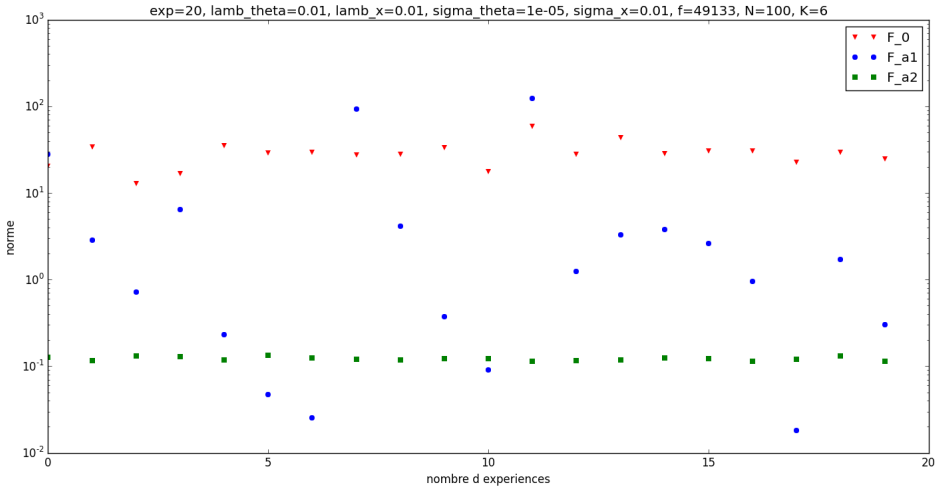

Figure 12: Comparaison des normes de F_0, F_{a_1}, F_{a_2}

On remarque tout d'abord que l'algorithme sans contrainte ne minimise pas correctement F tout le temps puisque pour certaines conditions initiales $F_{a_1} \geqslant F_0$. En ajoutant la contrainte, la norme de F est constamment divisée par plus d'un facteur 100. On remarque une très bonne stabilité des résultats avec terme forçant.

Regardons ce que cela donne concernant les distances aux points réels :

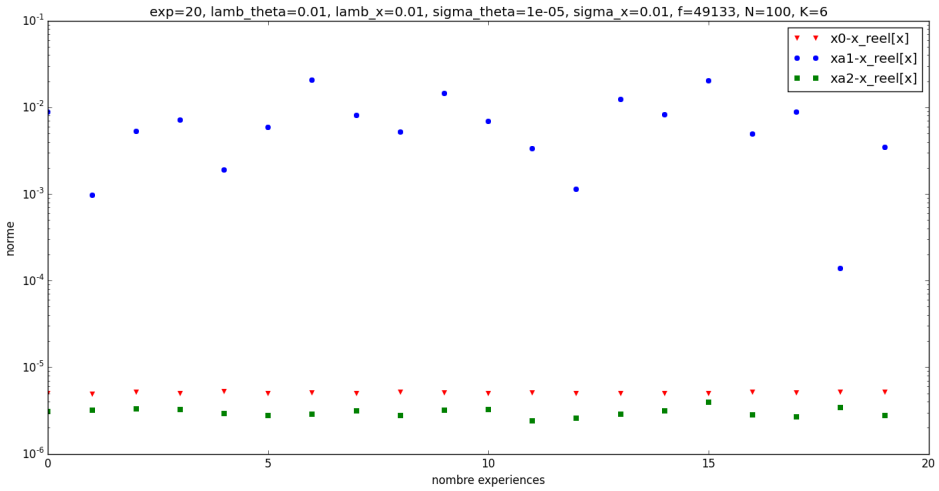

Figure 13: Comparaison des erreurs selon x

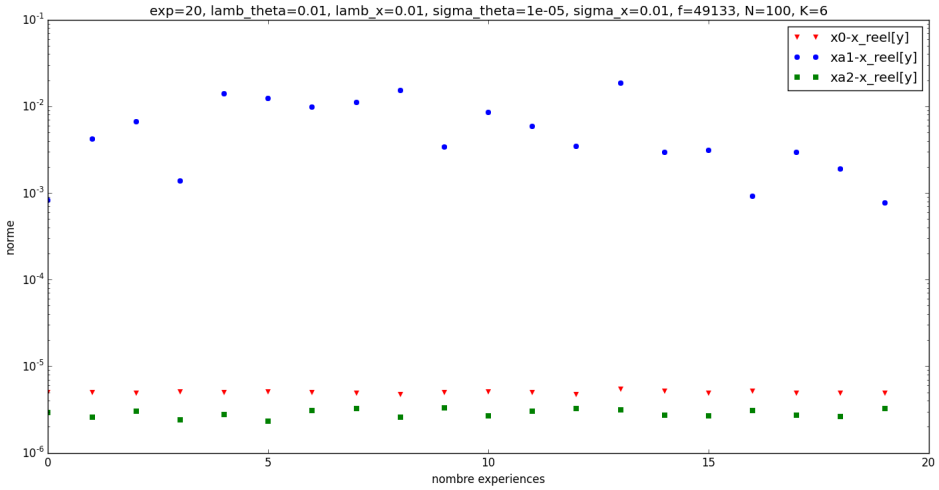

Figure 14: Comparaison des erreurs selon y

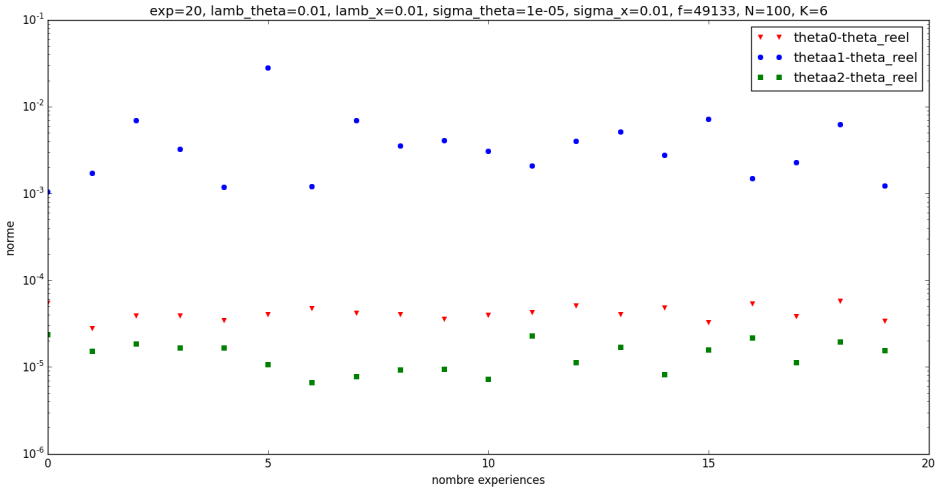

Figure 15: Comparaison des erreurs sur les angles

La méthode avec contrainte nous permet d'obtenir des résultats bien plus satisfaisants qu'avant : on divise en moyenne l'erreur initiale sur les angles et les coordonnées des points par 2, là où l'on s'éloignait loin de la solution réelle avec la méthode sans contrainte.

Remarque On peut alors essayer de procèder à plusieurs itérations de l'algorithme, mais on remarque que cela n'améliore pas beaucoup plus la solution.

5.1.4 Interprétation

Nos observations nous assurent tout d'abord que notre méthode d'approximation des angles des caméras et des coordonnées des points fonctionne : notre algorithme nous fournit une solution qui est plus proche de la solution réelle que la solution initiale.

Dans la figure 6, la régularité de la norme de F_{a2} et l'extrême irrégularité de la norme de F_{a1} laissent penser

que la matrice M' (pour le problème avec contrainte) est bien mieux conditionnée que la matrice M (pour le problème sans contrainte).

6 Comparaison avec les méthodes classiques de Bundle Adjustment

Afin de mieux comprendre l'intérêt de notre méthode et de nos hypothèses par rapport aux techniques classiques de bundle adjustment, nous avons implémenté l'algorithme classique afin de comparer les résultats des deux méthodes.

[[2] ?]

References

[1] Richard HARTLEY and Andrew ZISSERMAN. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[2] Pierre MOULON, Pascal MONASSE, Renaud MARLET, and Others. OpenMVG. `https://github.com/openMVG/openMVG`.