Chapitre 8

Localisation caméra/objet

Dans ce chapitre on s'intéresse au problème de la détermination de la position et l'orientation d'un objet par rapport à une caméra. Ce problème est équivalent au problème de la détermination des paramètres extrinsèques d'une caméra lorsque les paramètres intrinsèques sont connus. Cependant il diffère du problème de calibrage car on n'a pas toujours à sa disposition un grand nombre de points de correspondance. En effet, un grand nombre de points de correspondance permet une approche linéaire qui est bien conditionnée. Lorsqu'il y a peu de points de correspondance l'approche linéaire est mal conditionnée. Nous allons donc introduire une technique non linéaire d'estimation des paramètres recherchés.

8.1. Introduction

Plus formellement, le problème est le suivant : étant donné un ensemble de points ou de droites 3D décrits dans un référentiel "objet" et leurs projections 2D décrites dans un référentiel "caméra" connaissant les paramètres de la caméra, il faut alors déterminer la transformation rigide (rotation et translation) entre le référentiel objet et le référentiel caméra.

Les solutions proposées pour résoudre ce problème peuvent se grouper en 2 catégories :

1. *solutions analytiques.* Lorsque le nombre de points de correspondance est petit (entre 3 et 6 correspondances) on peut trouver une solution analytique qui consiste essentiellement à résoudre un système d'équations non linéaires. Cependant, pour 3 points, il peut y avoir jusqu'à 4 solutions [62]. Des solutions

existent également pour 4 points coplanaires [101] ou pour 4 points non coplanaires [90], [86]. On a également proposé des solutions pour 3 droites [48], [33], ce qui est équivalent à 3, 4, 5 ou 6 points comme il est montré sur la figure 8.2.

2. *solutions numériques.* Lorsque le nombre de points de correspondance est grand, les solutions analytiques ne sont plus efficaces, car on doit résoudre un système d'équations non linéaires où le nombre d'équations est grand par rapport au nombre d'inconnues. Dans ce cas, des solutions numériques sont nécessaires. Nous allons discuter de quelques solutions proposées.

Comme nous allons le voir, le problème étant non linéaire, les choix relatifs quant à la formulation mathématique, quant à l'expression de la fonction d'erreur à minimiser et quant à la méthode de minimisation, sont cruciaux.

Ganapathy [66] présente une solution linéaire, la rotation étant représentée par une matrice 3×3, sans exprimer explicitement les contraintes d'orthonormalité de cette matrice. Cette méthode est extrêmement sensible au bruit.

Yuan [180] a proposé de séparer la rotation de la translation et il a concentré ses efforts sur la détermination de la rotation. Les six contraintes d'orthonormalité de la matrice de rotation donnent naissance à 6 contraintes quadratiques. La solution commune à ces 6 contraintes est trouvée utilisant la méthode de Newton (descente de gradient). L'auteur a remarqué que des minima locaux peuvent être trouvés avec cette méthode. Plusieurs minima locaux correspondent à la nature non linéaire du problème. Le minimum global peut être atteint dans ce cas, seulement si on fournit une initialisation proche de la solution. Lowe [114] a également utilisé la méthode de Newton. La fonction d'erreur à minimiser est la somme des carrés des distances entre chaque projection d'un point (ou d'une droite) de l'objet et son correspondant dans l'image. Comme dans le cas de la méthode de Yuan, Lowe a rencontré des problèmes avec cette méthode et dans un autre article, il a suggéré des modifications de la méthode pour résoudre les problèmes d'initialisation et de stabilité [115]. La méthode de stabilisation qu'il a suggérée n'est efficace qu'à condition que certains des paramètres soient correctement initialisés.

Liu et al. [111] ont examiné une méthode itérative, la rotation étant représentée par les angles d'Euler. D'abord, ils constatent que, si on met en correspondance des droites et non pas des points, la rotation est séparée de la translation et qu'une fois que l'on a déterminé la rotation, la détermination de la translation devient un problème linéaire. Cependant les auteurs constatent que leur méthode n'est efficace que lorsque les angles de la rotation sont inférieurs à 30^0.

Utilisant la même représentation que Liu et al., Kumar et Hanson [108] ont examiné deux méthodes de minimisation : une technique itérative qui linéarise la fonction d'erreur et une technique combinatoire. Cette dernière technique permet, en principe, d'éliminer des mesures aberrantes mais elle est très lente, car elle considère tous les triplets possibles de correspondance.

Dans ce qui suit nous allons considérer des correspondances de droites et de points. Utilisant les quaternions unitaires pour décrire les rotations, les contraintes du problème deviennent quadratiques. La fonction d'erreur sera la somme des carrés de ces contraintes quadratiques. Nous allons ensuite introduire une méthode d'optimisation non linéaire dite de *région de confiance* [179], [150] ainsi qu'une méthode linéaire utilisant un modèle rapproché de caméra et un algorithme itératif. En effet, toute méthode de minimisation non-linéaire nécessite une initialisation de la solution. Plus cette solution initiale est proche de la solution recherchée, plus rapide et plus sure sera la convergence de l'algorithme de minimisation non linéaire.

8.2. Correspondances de droites

Nous considérons le modèle géométrique de caméra tel qu'il a été développé au chapitre 5. Un point m de l'image, qui a u et v comme coordonnées dans le repère image, aura x_c et y_c comme coordonnées caméra :

$$\begin{aligned} x_c &= (u - u_0)/\alpha_u \\ y_c &= (v - v_0)/\alpha_v \end{aligned}$$

où u_0, v_0, α_u, et α_v sont les paramètres intrinsèques de la caméra. Soient x, y et z les coordonnées dans le repère caméra d'un point 3D M se trouvant sur un objet et se projetant dans l'image en m. On a :

$$\left\{ \begin{array}{l} x_c = \frac{x}{z} \\ y_c = \frac{y}{z} \end{array} \right. \qquad [8.1]$$

De plus, nous allons contraindre le point m d'appartenir à une droite dans l'image dont l'équation s'écrit :

$$a x_c + b y_c + c = 0 \qquad [8.2]$$

En substituant l'équation [8.1] dans l'équation [8.2] on obtient :

$$ax + by + cz = \mathbf{n} \cdot \overrightarrow{FM} = 0 \qquad [8.3]$$

ceci n'est autre que l'équation du plan contenant le centre de projection, la droite de l'image et le point M ; $\mathbf{n}$ est le vecteur normal à ce plan, comme sur la figure 8.1.

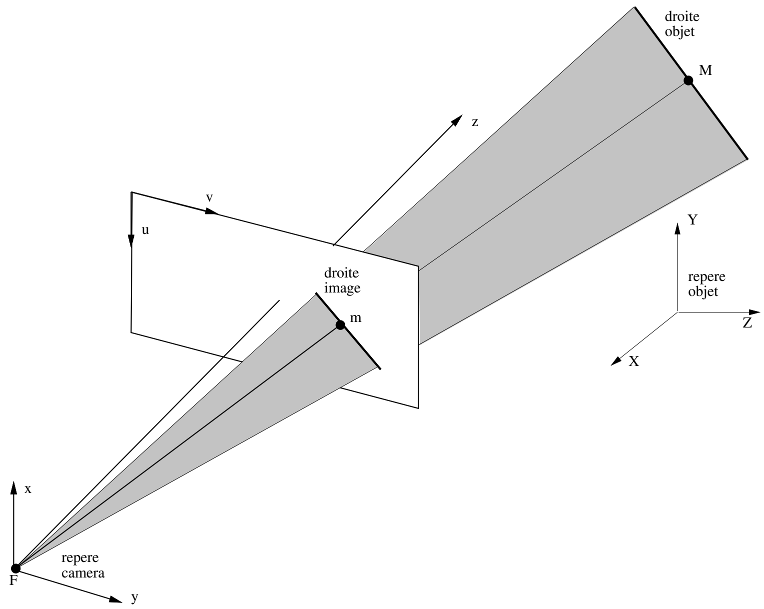

Figure 8.1. *La droite "objet", sa projection dans l'image et le centre de projection F sont coplanaires et ce plan est montré en gris. $\mathbf{n}$ est le vecteur unitaire orthogonal à ce plan.*

Considérons maintenant la droite objet décrite par le point M. Dans le référentiel de l'objet, cette droite peut être décrite par sa représentation paramétrique. Soient $\mathbf{v}$ son vecteur directeur (unitaire) et $\mathbf{p}$ son vecteur position. Cette droite peut également être exprimée dans le repère de la caméra :

$$\overrightarrow{FM} = \mathbf{p}' + \mu \mathbf{v}'$$

La relation entre les paramètres de la droite dans deux repères différents étant :

$$\begin{aligned} \mathbf{p}' &= \mathbf{R}\ \mathbf{p} + \mathbf{t} \\ \mathbf{v}' &= \mathbf{R}\ \mathbf{v} \end{aligned}$$

où $\mathbf{R}$ et $\mathbf{t}$ décrivent la rotation et la translation entre le référentiel objet et le référentiel caméra et contiennent justement les paramètres extrinsèques que l'on se propose de déterminer. La contrainte de correspondance exprime le fait que la droite objet appartient au plan défini auparavant, équation [8.3] :

$$\begin{cases} \mathbf{n} \cdot \mathbf{v}\,' = 0 \\ \mathbf{n} \cdot \mathbf{p}\,' = 0 \end{cases}$$

ce qui peut s'écrire :

$$\begin{cases} \mathbf{n} \cdot (\mathbf{R}\ \mathbf{v}) = 0 \\ \mathbf{n} \cdot (\mathbf{R}\ \mathbf{p} + \mathbf{t}) = 0 \end{cases} \qquad [8.4]$$

Par conséquent, chaque correspondance droite-objet/droite-image fournit 2 contraintes. Puisque le nombre de paramètres inconnus qu'il faut déterminer est 6 (3 pour la rotation et 3 pour la translation), le problème peut être résolu si au moins 3 correspondances sont disponibles. En définitive, 3 droites peuvent être construites grâce à 3, 4, 5, ou 6 points, comme sur la figure 8.2. La solution que nous présentons dans ce chapitre est donc valable pour des correspondances de droites et de points. Il faut avoir au minimum 3 correspondances point/point ou 3 correspondances droite/droite. Dans le cas général, lorsque N correspondances droite/droite sont disponibles, le problème est de résoudre $2N$ contraintes non linéaires, ce qui est équivalent au problème de minimiser la fonction d'erreur suivante :

$$f(\mathbf{R}\ , t) = \sum_{i=1}^{N} (\mathbf{n}_i \cdot (\mathbf{R}\ \mathbf{v}_i))^2 + \sum_{i=1}^{N} (\mathbf{n}_i \cdot (\mathbf{R}\ \mathbf{p}_i + \mathbf{t}))^2 \qquad [8.5]$$

8.3. La fonction d'erreur

Grâce à la représentation des rotations par des quaternions unitaires (section 7.5.6.), nous pouvons transformer la fonction d'erreur que l'on vient d'établir en une somme de carrés de contraintes quadratiques. Une approche analogue utilisant des quaternions duaux pour représenter les rotations et translations est proposée dans [150]. Rappelons que les vecteurs de dimension 3 vont être traités comme des quaternions purement imaginaires. La première contrainte de l'équation [8.4] peut s'écrire de la manière suivante :

$$\begin{aligned} \mathbf{n} \cdot (\mathbf{R}\ \mathbf{v}) &= \mathbf{n}^t (W(\mathbf{r})^t Q(\mathbf{r}) \mathbf{v}) \\ &= \mathbf{r}^t Q(\mathbf{n})^t W(\mathbf{v}) \mathbf{r} \end{aligned} \qquad [8.6]$$

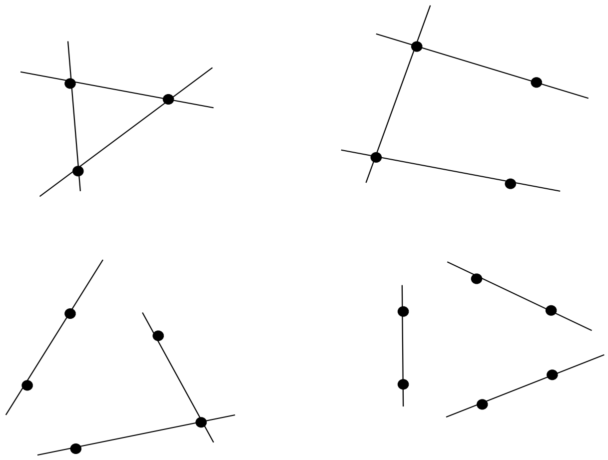

Figure 8.2. *Configurations de 3, 4, 5 ou 6 points qui peuvent être remplacées par 3 droites.*

où $\mathbf{r}$ est le quaternion unitaire associé à la matrice de rotation $\mathbf{R}$. La seconde contrainte de l'équation [8.4] devient :

$$\begin{aligned} \mathbf{n} \cdot (\mathbf{R}\ \mathbf{p} + \mathbf{t}) &= \mathbf{n}^t (W(\mathbf{r})^t Q(\mathbf{r})\mathbf{p} + \mathbf{t}) \\ &= \mathbf{r}^t Q(\mathbf{n})^t W(\mathbf{p})\mathbf{r} + \mathbf{n}^t \mathbf{t} \end{aligned} \qquad [8.7]$$

En d'autres termes, chaque correspondance droite/droite i fournit 2 contraintes quadratiques :

$$\mathbf{r}^t A_i \mathbf{r} = 0$$

et :

$$\mathbf{r}^t B_i \mathbf{r} + \mathbf{n}_i^t \mathbf{n} = 0$$

avec la contrainte $\mathbf{r}^t\mathbf{r} = 1$ et A_i, B_i étant 2 matrices $4{\times}4$ définies par :

$$\begin{aligned} A_i &= Q(\mathbf{n}_i)^t W(\mathbf{v}_i) \\ B_i &= Q(\mathbf{n}_i)^t W(\mathbf{p}_i) \end{aligned}$$

On peut maintenant écrire une nouvelle expression pour la fonction d'erreur associée à notre problème, soit l'équation [8.5] :

$$f(\mathbf{r},\ \mathbf{t}) = \sum_{i=1}^{N} ((\mathbf{r}^t A_i \mathbf{r})^2 + (\mathbf{r}^t B_i \mathbf{r} + \mathbf{n}_i^t \mathbf{t})^2) + \lambda(\mathbf{r}^t\mathbf{r} - 1)^2 \qquad [8.8]$$

où les paramètres à estimer sont les éléments du quaternion unitaire $\mathbf{r}$ et du vecteur de translation $\mathbf{t}$. Le dernier terme de la fonction d'erreur est un terme de pénalisation avec $\lambda > 0$. Plus λ sera grand et mieux on garantira que le quaternion sera unitaire. En pratique, la valeur de λ peut être comprise entre 50 et 100. Une valeur trop grande de λ peut considérablement ralentir la vitesse de convergence de l'algorithme d'optimisation.

On peut noter qu'une alternative consisterait à estimer la rotation et la translation séparément. On peut estimer tout d'abord la rotation en utilisant la fonction d'erreur suivante :

$$f(\mathbf{r}) = \sum_{i=1}^{N} (\mathbf{r}^t A_i \mathbf{r})^2 + \lambda(\mathbf{r}^t\mathbf{r} - 1)^2 \qquad [8.9]$$

Une fois que l'on a estimé la rotation optimale, le calcul de la translation optimale est un problème linéaire.

8.4. Correspondance de points

La fonction d'erreur que nous venons d'établir est valable pour des correspondances de droites ET pour des correspondances de points puisque un ensemble de 2 contraintes quadratiques peuvent être associées soit avec une droite soit avec 2 points. Dans cette section nous allons établir une fonction d'erreur directement à partir de correspondances de points : 2 contraintes pour chaque correspondance point/point. Comme dans le cas précédent, ces contraintes sont quadratiques et elles décrivent la colinéarité entre le point 3D, sa projection 2D et le centre de projection du modèle de la caméra.

Comme auparavant, on considère un point M_i de l'objet ayant comme coordonnées X_i, Y_i et Z_i dans un repère lié à l'objet, et soient x_i et y_i les coordonnées dans le repère de la caméra de sa projection m_i. La colinéarité entre M_i, m_i, et F (le centre de projection) s'exprime avec les deux équations suivantes :

$$x_i = \frac{r_{11}X_i + r_{12}Y_i + r_{13}Z_i + t_1}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_3} \qquad [8.10]$$

$$y_i = \frac{r_{21}X_i + r_{22}Y_i + r_{23}Z_i + t_2}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_3} \qquad [8.11]$$

Dans ces équations r_{ij} est un éléement de la matrice de rotation $\mathbf{R}$ et t_i est une composante du vecteur de translation $\mathbf{t}$. Ces équations peuvent également s'écrire sous la forme matricielle suivante :

$$\begin{pmatrix} 1 & 0 & -x_i \end{pmatrix} \mathbf{R} \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + \begin{pmatrix} 1 & 0 & -x_i \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \quad [8.12]$$

$$\begin{pmatrix} 0 & 1 & -y_i \end{pmatrix} \mathbf{R} \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + \begin{pmatrix} 0 & 1 & -y_i \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \quad [8.13]$$

Soit $\mathbf{M}_i$ le quaternion purement imaginaire associé au vecteur point M_i et soient $\mathbf{m}_{x_i}$ et $\mathbf{m}_{y_i}$ deux quaternions purement imaginaires, chacun étant associé avec une “moitié” du vecteur point m_i :

$$\begin{aligned} \mathbf{M}_i &= (0\ X_i\ Y_i\ Z_i)^t \\ \mathbf{m}_{x_i} &= (0\ 1\ 0\ -x_i)^t \\ \mathbf{m}_{y_i} &= (0\ 0\ 1\ -y_i)^t \end{aligned}$$

Si on remplace la matrice de rotation $\mathbf{R}$ par sa représentation en terme d'un quaternion unitaire (voir la section 7.5.6.), on obtient pour la première contrainte :

$$\mathbf{m}_{x_i}^t W(\mathbf{r})^t Q(\mathbf{r}) \mathbf{M}_i + \mathbf{m}_{x_i}^t \mathbf{t} = \mathbf{r}^t Q(\mathbf{m}_{x_i})^t W(\mathbf{M}_i) \mathbf{r} + \mathbf{m}_{x_i}^t \mathbf{t}$$

Il y a une expression similaire pour la deuxième équation :

$$\mathbf{m}_{y_i}^t W(\mathbf{r})^t Q(\mathbf{r}) \mathbf{M}_i + \mathbf{m}_{y_i}^t \mathbf{t} = \mathbf{r}^t Q(\mathbf{m}_{y_i})^t W(\mathbf{M}_i) \mathbf{r} + \mathbf{m}_{y_i}^t \mathbf{t}$$

Finalement, les équations [8.10] et [8.11] deviennent :

$$\mathbf{r}^t C_i \mathbf{r} + \mathbf{m}_{x_i}^t \mathbf{t} = 0 \quad [8.14]$$

$$\mathbf{r}^t D_i \mathbf{r} + \mathbf{m}_{y_i}^t \mathbf{t} = 0 \quad [8.15]$$

Dans ces équations les matrices 4×4 C_i et D_i sont données par :

$$C_i = Q(\mathbf{m}_{x_i})^t W(\mathbf{M}_i)$$

$$D_i = Q(\mathbf{m}_{y_i})^t W(\mathbf{M}_i)$$

La fonction d'erreur dans le cas des points s'écrit comme la somme au carré de ces fonctions quadratiques sous la contrainte de quaternion unitaire pour $\mathbf{r}$:

$$f(\mathbf{r}, \mathbf{t}) = \sum_{i=1}^{N} \left((\mathbf{r}^t C_i \mathbf{r} + \mathbf{m}_{x_i}^t \mathbf{t})^2 + (\mathbf{r}^t D_i \mathbf{r} + \mathbf{m}_{y_i}^t \mathbf{t})^2 \right) + \lambda (\mathbf{r}^T \mathbf{r} - 1)^2 \quad [8.16]$$

8.5. Optimisation : la méthode de région de confiance

La minimisation de la fonction d'erreur décrite par les équations [8.8] et [8.9] est équivalente à celle du critère quadratique non linéaire suivant :

$$0 = \min\{f(x) = \frac{1}{2}\sum_{j=1}^{m}\Phi_j^2(x) : x \in \mathbb{R}^n\} \qquad [8.17]$$

Dans cette expression les $\Phi_j(x)$ sont des fonctions continues et deux fois dérivables de $\mathbb{R}^n$ dans $\mathbb{R}$. Dans notre cas x et soit de dimension 4 ($\mathbf{r}$) soit de dimension 7 ($\mathbf{r}$ et $\mathbf{t}$). On rappelle, par ailleurs, que le gradient $\nabla f(x)$ et le hessien $\nabla^2 f(x)$ peuvent être calculés comme suit :

$$\begin{aligned} g(x) = \nabla f(x) &= \sum_{j=1}^{m}\Phi_j(x)\nabla\Phi_j(x) = J(x)^t\Phi(x), && [8.18] \\ H(x) = \nabla^2 f(x) &= \sum_{j=1}^{m}\nabla\Phi_j(x)\nabla\Phi_j(x)^t + \sum_{j=1}^{m}\Phi_j(x)\nabla^2\Phi_j(x) \\ &= J(x)^t J(x) + \sum_{j=1}^{m}\Phi_j(x)\nabla^2\Phi_j(x) && [8.19] \end{aligned}$$

où $\Phi(x) = (\Phi_1(x), \ldots, \Phi_m(x))^t$ et $J(x) = (\nabla\Phi_1(x), \ldots, \nabla\Phi_m(x))^t$ est la matrice jacobienne $m \times n$ de $\Phi(x)$. On peut facilement voir que $\Phi(x)$ est deux fois continue et différentiable de $\mathbb{R}^n$ dans $\mathbb{R}^m$ et que $J(x)$ est une matrice non singulière. Le hessien d'une telle fonction quadratique est une combinaison de termes du premier et deuxième ordre :

$$H(x) = J(x)^t J(x) + Q(x)$$

avec : $Q(x) = \sum_{j=1}^{m}\Phi_j(x)\nabla^2\Phi_j(x)$. En pratique l'approximation Gauss-Newton du hessien est utilisée, soit $H(x) = J(x)^t J(x)$. Cela est basé sur l'hypothèse que le terme du premier ordre va finalement dominer le terme du deuxième ordre $Q(x)$ [70].

On désigne par x_k l'estimation courante de la solution. Une variable avec l'indice k désigne cette variable évaluée à la k^e itération de l'algorithme. L'idée de base de la méthode de région de confiance est d'approcher successivement la fonction d'erreur par une *forme quadratique locale* dans un voisinage de la solution courante x_k :

$$f(x_k + d) \approx f(x_k) + q_k(d), \ \ ||d|| \leq \delta_k$$

où :

$$q_k(d) = g(x_k)^t d + \frac{1}{2} d^t H(x_k) d \qquad [8.20]$$

La fonction d'erreur sera réduite selon la direction d_k, soit $x_{k+1} = x_k + d_k$, où d_k est la valeur de d minimisant la forme quadratique locale sur une région sphérique centrée sur x_k : cette sphère s'appelle la *région de confiance* et on a :

$$d_k = \min\{q_k(d) : ||d|| \leq \delta_k\} \qquad [8.21]$$

Le paramètre δ_k est le rayon de confiance et il est déterminé dynamiquement en utilisant une mesure de la qualité de l'approximation. Cette qualité est mesurée par un coefficient de qualité r_k :

$$r_k = \frac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)} \qquad [8.22]$$

mesurant le rapport entre la *vraie réduction* de la fonction d'erreur lorsque l'on se déplace de x_k à $x_k + d_k$ et la *réduction prédite* par l'approximation quadratique. Si r_k est trop petit, cela veut dire que l'approximation n'est pas bonne et la taille de la région de confiance doit être réduite. Sinon, la région de confiance doit être augmentée en taille. Schématiquement, la stratégie suivante sera appliquée :

- si $r_k \geq r$ (avec $0 < r \leq 1$), alors l'itération doit être acceptée et $x_{k+1} = x_k + d_k$. De plus, la taille de la région de confiance est augmentée.

- si $r_k < r$, alors l'itération n'est pas acceptée et la taille de la région de confiance doit être diminuée. Ce processus est répété jusqu'à ce que la forme quadratique locale fournisse une approximation satisfaisante à l'intérieur de la région de confiance – ce qui va nécessairement se passer pour des rayons petits, car le gradient est connu et le terme du premier ordre de q_k devient dominant lorsque $||q_k|| \neq 0$.

La forme quadratique locale dépend du gradient et du hessien de la fonction d'erreur. Ceci implique que le minimum ainsi trouvé a de "bonnes" propriétés du second ordre. La majeure difficulté de la méthode de région de confiance réside dans la minimisation de la forme quadratique locale. Les quelques versions de la méthode de région de confiance diffèrent par la méthode qui est utilisée pour minimiser la forme quadratique locale. La méthode que nous avons choisi d'utiliser est décrite en détail dans la section suivante.

Nous pouvons comprendre la différence entre les méthodes classiques de minimisation non linéaire et la méthode de région de confiance grâce à un exemple de minimisation d'une fonction $f(x)$ à une variable. Une telle fonction est illustrée sur la figure 8.3. A partir d'un point initial x^0, une méthode du type Newton approxime la fonction avec une parabole utilisant un développement limité au deuxième ordre autour de x^0. Le point suivant, x^1 correspond au minimum de cette parabole mais cette valeur augmente la fonction $f(x)$. L'idée de la méthode de région de confiance est d'approximer la fonction avec une parabole également et de chercher le minimum sur cette parabole dans une zone restreinte plutôt que sur tout le domaine de définition de la fonction à minimiser. La figure 8.4. montre le comportement d'une telle méthode sur la même fonction. A partir du point x^0 le minimum de la parabole est recherché sur l'intervalle de taille $2r_0$. Le minimum est dans ce cas une des extrémités de la parabole : $x^1 = x^0 - r_0$. Autour de x^1 la fonction est à nouveau approximée par une parabole mais, cette fois-ci, sur un intervalle 2 fois plus large : $r_1 = 2r_0$. Le minimum de cette nouvelle parabole est en $x^2 = x^1 + r_1$. A partir de maintenant le comportement de l'algorithem de région de confiance est très similaire à la méthode de Newton et il converge vers le point x^*.

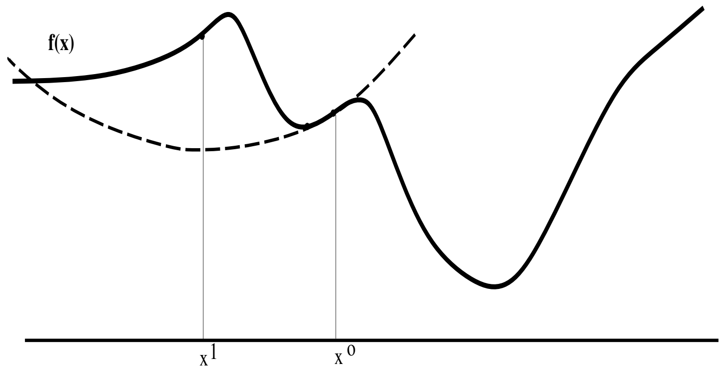

Figure 8.3. *Un exemple de minimisation lorsque plusieurs minima sont présent. Dans ce cas, la méthode de minimisation de Newton peut conduire à l'augmentation de la valeur de la fonction $f(x)$.*

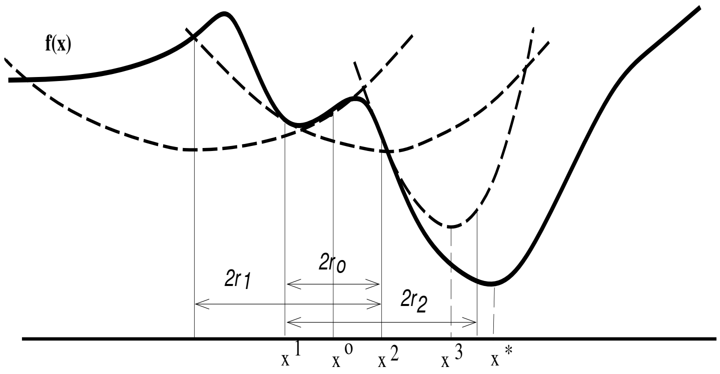

Figure 8.4. *Cette figure montre un exemple de minimisation de la fonction $f(x)$ avec la méthode de région de confiance. Cette méthode garantit que la valeur de la fonction à minimiser décroit.*

8.5.1. Minimisation de la forme quadratique locale

Il s'agit du problème de minimisation d'une forme quadratique à l'intérieur d'une boule :

$$\min\{\frac{1}{2}d^t H d + g^t d : ||d|| \leq \delta\} \qquad (LQP)$$

où g est un vecteur, H est une matrice symétrique et δ est un nombre positif. Puisque l'ensemble de définition est un compact, le problème a une solution. Toutes les méthodes existantes pour résoudre ce problème sont basées sur le théorème suivant ([69] et [137]) :

Théorème 8.1 *d^* est une solution de (LQP) si et seulement s'il existe $\mu \geq 0$ tel que :*

(i) $H + \mu I$ est positive semi-définie,

(ii) $(H + \mu I)d^* = -g$,

(iii) $||d^*|| \leq \delta$ et $\mu(||d^*|| - \delta) = 0$.

Un tel μ est unique.

Si μ peut être estimé, la solution optimale de notre problème est facile à calculer. En fait, la minimisation de (LQP) représente un cas intéressant d'optimisation non convexe et, dans ce qui suit, nous allons donner une caractérisation complète de la solution. La solution elle-même sera donnée par un algorithme dont nous allons donner la trame.

Pour $\mu > 0$, le problème (LQP) a une solution sur la frontière de son ensemble de définition, soit sur la frontière de la région de confiance :

$$\|d^*\| = \delta$$

et le problème se réduit à celui de trouver μ tel que :

$$\phi(\mu) = \|d(\mu)\| = \delta \qquad [8.23]$$

où $d(\mu)$ est solution de $(H + \mu I)d = -g$. Soient $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ les valeurs propres de H (une matrice symétrique) et soient $u_1, u_2, \ldots, u_n$ les vecteurs propres correspondants. La fonction ϕ peut être écrite explicitement en utilisant le système propre de H :

$$\phi(\mu) = \left(\sum_{i=1}^{n} \frac{(u_i^t g)^2}{(\lambda_i + \mu)^2} \right)^{1/2} \qquad [8.24]$$

et on peut vérifier les propriétés suivantes de ϕ :

(i) $\phi(\mu)$ est positive, strictement décroissante sur l'intervalle $]-\lambda_1, +\infty[$ et $\phi(\mu) \to 0$ lorsque $\mu \to \infty$.

(ii) $\phi^2(\mu)$ est une fonction rationnelle de μ, avec des pôles du deuxième ordre sur un sous-ensemble de l'ensemble $\{-\lambda_1, -\lambda_2, \ldots -\lambda_n\}$ des opposés des valeurs propres de H [84].

Hebden [84] a proposé un algorithme pour déterminer μ et d^* (la solution optimale) itérativement lorsque l'approximation Gauss-Newton du hessien est suffisante. Dans ce cas, H est symétrique, semi-définie et positive. Hebden a proposé de mettre à jour μ_{l+1} en fonction de μ_l comme suit :

$$\mu_{l+1} = \mu_l + \frac{\phi(\mu_l)}{\phi'(\mu_l)} \frac{(\delta - \phi(\mu_l))}{\delta}. \qquad [8.25]$$

En fait, l'algorithme de Hebden peut être vu comme une méthode de Newton pour trouver la solution de l'équation :

$$\psi(\mu) = \frac{1}{\delta} - \frac{1}{\phi(\mu)} = 0, \quad \text{pour } \mu \in]-\lambda_1, +\infty[\qquad [8.26]$$

La caractéristique la plus importante de l'équation [8.25] est que le nombre d'itérations normalement nécessaire pour aboutir à une bonne approximation de la solution μ^* est petit, car ψ est convexe, presque linéaire et strictement décroissant sur l'intervalle $]-\lambda_1, +\infty[$. De plus, le calcul de la factorisation de Cholesky de $H + \mu I$ rend possible le calcul décrit par l'équation [8.25] sans connaissance explicite des éléments propres de H, dès lors que $\mu \in]-\lambda_1, +\infty[$. Plus précisément, il est facile de voir que :

$$\phi(\mu) = ||d(\mu)||, \qquad \phi'(\mu) = -[1/\phi(\mu)]d(\mu)^t(H + \mu I)^{-1}d(\mu),$$

où $(H+\mu I)d(\mu) = -g$. Donc, si R^tR est la factorisation de Cholesky de $H+\mu_l I$, où R est une matrice triangulaire supérieure, alors nous avons :

$$\frac{\phi(\mu_l)}{\phi'(\mu_l)} = -\frac{\phi^2(\mu_l)}{d(\mu_l)^t(R^tR)^{-1}d(\mu_l)} = -\frac{\phi^2(\mu_l)}{||(R^t)^{-1}d(\mu_l)||^2}$$

En conclusion, l'algorithme de Hebden, qui trouve le minimum de la forme quadratique locale, peut être résumé comme suit :

- *initialisation* : Soit $\mu_o \geq 0$ et $\phi(\mu_o) > \delta$.
- *itération l=0,1,...*

l.1 factoriser $(H + \mu_l I) = R^tR$.

l.2 résoudre $R^tRp = -g$.

l.3 si $|||p|| - \delta| < \varepsilon$, alors stop: μ_l est une approximation de μ^* et p est une solution de (LQP).

l.4 sinon, résoudre $R^tq = p$.

l.5 incrémenter

$$\mu_{l+1} = \mu_l + \left(\frac{||p||}{||q||}\right)^2 \frac{(||p|| - \delta)}{\delta}. \qquad [8.27]$$

l.6 incrémenter l:=l+1 et retourner à l.1.

En pratique, cet algorithme converge après 4 itérations.

8.5.2. L'algorithme de région de confiance

Nous sommes maintenant en mesure de décrire l'algorithme de région de confiance qui est bien adapté au problème de détermination des paramètres extrinsèques. L'idée est d'utiliser l'approximation Gauss-Newton du hessien pour déterminer la direction de recherche à chaque itération. Dans ce cas, H est définie et positive, on a donc $\lambda_1 > 0$ et la condition (i) du théorème 8.1

est vraie pour tout $\mu \geq 0$. D'abord, on vérifie si $\mu = 0$ lors de la résolution de $H(x)d = -g$. Si $||d|| \leq \delta$, alors d est la solution optimale pour (LQP). Sinon, $\mu > 0$ et on peut appliquer l'algorithme de Hebden. La trame de l'algorithme de région de confiance est la suivante :

- *initialisation* : Soient

- x_o, valeur initiale.

- δ_o, rayon de confiance initial.

- ε , test "zéro" général.

- ε_g, zéro pour $||g||$.

- ε_f, zéro pour f.

- $k := 0$.

- *itération k:=0,1,...*

k.1 calculer $f_k = f(x_k), g_k = \nabla f(x_k)$ et H_k.

k.2 si $||g_k|| \leq \varepsilon_g$ ou $\delta_k \leq \varepsilon_\delta$ ou $f_k \leq \varepsilon_f$, alors stop: x_k est une solution.

k.3 soit d une solution du système

$$H_k d = -g_k$$

Si $||d|| < \delta_k - \varepsilon$ alors $d_k = d$. Sinon, utiliser l'algorithme de Hebden pour trouver $\mu > 0$ tel que la solution de $(H_k + \mu I)d = -g_k$ satisfasse $|||d|| - \delta_k| < \varepsilon$, et on a $d_k = d$.

k.4 calculer r_k et utilisant l'équation [8.22].

k.5 si $r_k \geq s$, alors

$$x_{k+1} := x_k + d_k$$

si $r_k \geq t$, alors

$$\delta_{k+1} := 2\delta_k$$

sinon

$$\delta_{k+1} := \delta_k$$

Faire $k := k + 1$ et retourner à k.1

k.6 si $r_k < s$ alors

$$\delta_k := \delta_k / 2$$

et retourner à k.3.

Ceci est l'algorithme général de région de confiance. Le paramètre s doit appartenir à l'intervalle $[0.1, 0.3]$ et le paramètre t doit appartenir à l'intervalle $[0.5, 0.8]$, [179]. Dans notre cas, les valeurs de ces paramètres sont : $s = 0.25$ et $t = 0.75$.

8.6. Méthode linéaire avec une caméra affine

La méthode de minimisation non linéaire que nous venons de décrire nécessite une initialisation. Bien que l'algorithme de région de confiance ait des qualités de convergence qui sont meilleures que celles d'une simple méthode de Newton, il est important de pouvoir l'initialiser correctement afin de réduire le nombre d'itérations et s'assurer un temps de calcul réduit. Pour ce faire, nous allons introduire deux méthodes de calcul des paramètres de rotation et translation :

- une méthode linéaire qui utilise un modèle affine de caméra (voir la section 5.6.) et
- une méthode quasi linéaire qui utilise un algorithme itératif transformant le modèle affine de caméra en un modèle projectif.

Dans ce qui suit, nous allons traité le cas des correspondances de points – l'extension à des correspondances de droites est relativement aisée. Soit à nouveau les équations [8.10] et [eq:y-collinearity] de la projection d'un point M_i d'un objet sur l'image :

$$x_i = \frac{\mathbf{r}_1 \cdot \mathbf{M}_i + t_1}{\mathbf{r}_3 \cdot \mathbf{M}_i + t_3} \quad [8.28]$$

$$y_i = \frac{\mathbf{r}_2 \cdot \mathbf{M}_i + t_2}{\mathbf{r}_3 \cdot \mathbf{M}_i + t_3} \quad [8.29]$$

On divise les nominateurs et dénominateurs de ces fractions par t_3 et on obtient alors les equations suivantes :

$$x_i = \frac{\mathbf{I} \cdot \mathbf{M}_i + x_0}{1 + \varepsilon_i} \quad [8.30]$$

$$y_i = \frac{\mathbf{J} \cdot \mathbf{M}_i + y_0}{1 + \varepsilon_i} \quad [8.31]$$

Les nouvelles notations ont les significations suivantes :

- $\mathbf{I} = \mathbf{r}_1 / t_3$

- $\mathbf{J} = \mathbf{r}_2/t_3$
- $x_0 = t_1/t_3$ and $y_0 = t_2/t_3$ sont les coordonnées de la projection dans l'image de m_0 qui est la projection de M_0 – l'origine du repère objet
- $\varepsilon_i = \mathbf{r}_3 \cdot \mathbf{M}_i/t_3$.

On peut également écrire ces équations comme suit :

$$x_i(1+\varepsilon_i) - x_0 = \mathbf{I} \cdot \mathbf{M}_i \qquad [8.32]$$

$$y_i(1+\varepsilon_i) - y_0 = \mathbf{J} \cdot \mathbf{M}_i \qquad [8.33]$$

Lorsque le rapport entre la taille de l'objet et la distance de l'objet à la caméra est grand, on peut négliger la valeur de ε par rapport à 1. On obtient dans ce cas le modèle de perspective faible et les équations précédentes deviennent :

$$\begin{aligned} x_i - x_0 &= \mathbf{I} \cdot \mathbf{M}_i \\ y_i - y_0 &= \mathbf{J} \cdot \mathbf{M}_i \end{aligned}$$

Si l'on dispose d'au moins 4 points non coplanaires, on peut facilement calculer les vecteurs $\mathbf{I}$ et $\mathbf{J}$ de la façon suivante. Soit $\mathbf{x}$ et $\mathbf{y}$ les vecteurs formés avec les coordonnés image des points, et soit P la matrice $N \times 3$ formée avec les coordonnées 3D des points de l'objet. Les deux équations précédentes s'écrivent alors :

$$\begin{aligned} P\mathbf{I} &= \mathbf{x} \\ P\mathbf{J} &= \mathbf{y} \end{aligned}$$

Si les points de l'objet ne sont pas coplanaires, la matrice P est de rang égal à 3 et on peut alors calculer facilement $\mathbf{I}$ et $\mathbf{J}$:

$$\begin{aligned} \mathbf{I} &= (P^t P)^{-1} P^t \,\mathbf{x} \\ \mathbf{J} &= (P^t P)^{-1} P^t \,\mathbf{y} \end{aligned}$$

On en déduit facilement les paramètres de la rotation et translation entre le repère objet et le repère de la caméra :

$$\begin{aligned} t_3 &= \frac{1}{2}\left(\frac{1}{||\mathbf{I}||} + \frac{1}{||\mathbf{J}||}\right) \\ t_1 &= x_0\, t_3 \\ t_2 &= y_0\, t_3 \end{aligned}$$

$$\begin{aligned}\mathbf{r}_1 &= \frac{\mathbf{I}}{||\mathbf{I}||}\\ \mathbf{r}_2 &= \frac{\mathbf{J}}{||\mathbf{J}||}\\ \mathbf{r}_3 &= \mathbf{r}_1 \wedge \mathbf{r}_2\end{aligned}$$

8.7. Méthode quasi linéaire

La méthode linéaire qui vient d'être décrite permet une initialisation des paramètres lorsque l'on se trouve dans des conditions expérimentales proches du modèle affine de caméra. Que se passe-t-il si l'objet est relativement plus proche de la caméra. Dementhon [41] a proposé dans ce cas un algorithme qui applique le modèle précédent d'une manière itérative. On peut également étendre l'algorithme au cas des points coplanaires [142]. Une généralisation de ces algorithmes au cas de la paraperspective a été également proposée [89].

L'idée de base de tous ces algorithmes est de calculer itérativement des valeurs pour les ε_i dans les equations [8.32] et [8.33] jusqu'à l'obtention d'une solution stable. L'algorithme est le suivant :

1. pour tout i, $i \in \{1 \ldots N\}$, $N \geq 3$, $\varepsilon_i = 0$;

2. estimer les vecteurs $\mathbf{I}$ and $\mathbf{J}$,

3. calculer la position et l'orientation de l'objet par rapport à la caméra (voir les équations de la section précédente) ;

4. pour tout i, calculer:

$$\varepsilon_i = \frac{\mathbf{r}_3 \cdot \mathbf{M}_i}{t_3}$$

si les ε_i calculés à l'itération courante sont égaux à ceux calculés à l'itération précédente, arrêt de l'algorithme, sinon retourner à l'étape 2.

On peut se référer à [92] pour une analyse détaillé de cet algorithme ainsi que pour une étude expérimentale. Illustrons le comportement des algorithmes linéaire et quasi linéaire par les deux images de la figure 8.5.

8.8. Conclusion

Les méthodes que nous avons décrites dans ce chapitre sont particulièrement intéressantes lorsque l'on désire déterminer la position et l'orientation d'une

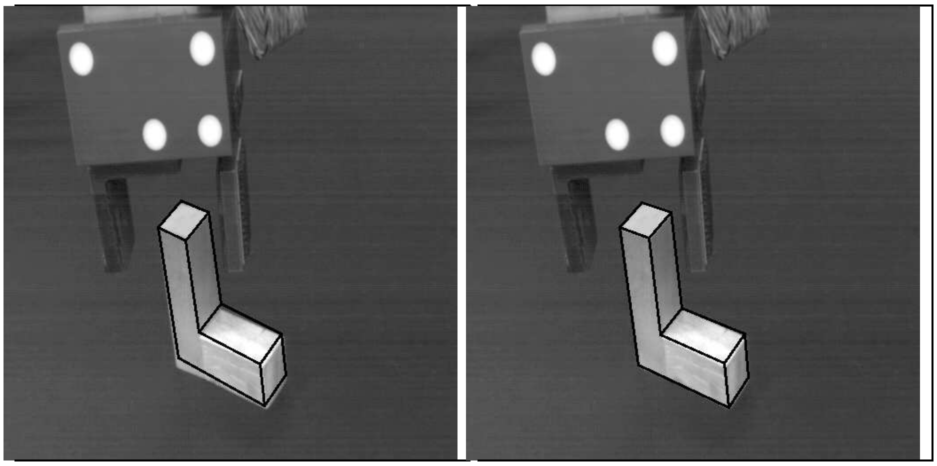

Figure 8.5. *Un exemple de calcul des paramètres d'orientation et position avec un algorithme linéaire (gauche) et avec un algorithme itératif, ou quasi linéaire (droite).*

caméra par rapport à un objet dans un contexte de reconnaissance d'objets. Ces méthodes doivent être distinguées des méthodes de calibration présentées au chapitre 5, qui mettaient l'accent sur l'obtention d'un modèle de capteur très précis en présence d'objets de calibrage spécialement construits à cet effet. Un certain nombre d'expérimentations décrites dans [149] et dans [150] permettent de comparer les résultats obtenus avec la méthode Faugeras-Toscani (chapitre 5 et [57]) avec la méthode de région de confiance décrite dans ce chapitre.

Une caractéristique importante de la méthode utilisant l'algorithme de région de confiance est que cette méthode n'est pas seulement robuste par rapport au bruit, mais également par rapport aux mesures aberrantes et par rapport aux erreurs d'appariement. La méthode linéaire du chapitre 5 nécessite au moins 6 points, alors que la méthode décrite ici peut en principe fonctionner avec seulement 3 points. En pratique la méthode de région de confiance est initialisée avec un algorithme linéaire ou avec un algorithme quasi linéaire et nécessite donc un minimum de 4 appariements.