INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

No. attribué par la bibliothèque

_ _ _ _ _ _ _ _ _

T H E S E

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : **Imagerie, Vision, Robotique**

préparée au laboratoire INRIA Rhône-Alpes / GRAVIR–IMAG

dans le cadre de l'Ecole Doctorale « **Mathématiques et Informatique** »

présentée et soutenue publiquement

par

Nicolas ANDREFF

le 29 novembre 1999

Titre :

Asservissement visuel à partir de droites et auto-étalonnage pince-caméra

---

Directeurs de thèse :

Bernard ESPIAU et Radu HORAUD

JURY

Mme Marie-Paule CANI , Présidente
M. François CHAUMETTE , Rapporteur
M. Bernard ESPIAU
M. Jean GALLICE , Rapporteur
M. Radu HORAUD
M. Hans-Helmut NAGEL

Avant-propos

Je tiens avant tout à remercier Marie-Paule Cani de m'avoir fait l'honneur de présider mon jury. Je suis aussi redevable à Jean Gallice et François Chaumette de leurs rapports constructifs et de leur patience (voire courage !) face au pavé que je leur ai fourni. J'associe enfin Hans-Helmut Nagel à ces remerciements, pour sa participation à mon jury et l'accueil qu'il m'a réservé pendant deux mois à Karlsruhe.

Je ne saurais dire à quel point furent fructueuses les discussions avec mes directeurs de thèse, Bernard Espiau et Radu Horaud, ni combien leur soutien et leurs conseils m'ont été précieux.

J'ai beaucoup apprécié l'accueil qui m'a été fait au sein de l'unité de recherche Rhône-Alpes de l'INRIA et du laboratoire GRAVIR de la fédération IMAG. En particulier, je me dois de mentionner l'accueil chaleureux de Roger Mohr dans les locaux de l'équipe MOVI.

J'ai aussi une pensée émue pour tous les BIPèdes et les MOVIens avec qui j'ai pu contribuer au progrès de la science dans la joie et la bonne humeur.

Merci encore à Éric Marchand pour le temps qu'il m'a consacré, en programmant le suiveur de trièdre et en m'aidant à obtenir les résultats expérimentaux d'asservissement visuel lorsque « mon » robot a décidé de se mettre en grève.

Merci à Roger « Président » et Hervé « Ronaldo », pour leur soutien logistique. Et encore pardon à toute l'équipe : c'est promis, je continuerai à vous crier après sur le terrain.

« *Cette* [thèse] *n'aurait pas été possible sans l'assistance permanente* [d'Isabelle] *qui a su et sait supporter les humeurs et les incontinences d'un chercheur éternellement obsédé par les problèmes majeurs de l'être, grâce à ses conseils rassérénants sur la vanité du tout. La constance avec laquelle elle m'a offert du jus de pomme en me le présentant comme un pur malt écossais des plus raffinés, a contribué de façon incommensurable et incroyable au fait que ces pages aient conservé un minimum de lucidité.* » (Umberto Eco [Eco97, p. 101])

Enfin, je ne saurais terminer sans exprimer ma profonde gratitude envers mes parents qui ont tout mis en œuvre pour me permettre de réaliser mon « *rêve sidéral d'un naïf idéal* ».

À Noëllie ...

Menu

Introduction

Friands Savoyards

Pour 4 personnes
Ingrédients : 100g de beurre, 150g de farine, 50cl de lait, noix de muscade, 2 jaunes d'œufs, 5cl de crème fraîche épaisse, 1 tranche de jambon cuit, 60g de fromage râpé, 2 œufs entiers, huile, chapelure, sel et poivre blanc

- Faire fondre le beurre dans une casserole. Verser la farine tamisée et remuer vivement avec un fouet. Faire fondre sur feu doux en remuant sans arrêt pendant 5 minutes. Lorsque le mélange est lisse, retirer la casserole du feu et verser le lait en continuant à fouetter régulièrement.

- Remettre la casserole sur feu doux et faire chauffer en remuant toujours pendant 10 minutes. Saler, poivrer et muscader. Incorporer les jaunes d'œufs et la crème fraîche, faire cuire pendant 5 minutes encore en remuant toujours sur feu doux.

- Retirer la casserole du feu. Tailler le jambon en petites lamelles puis en dés. Incorporer le jambon et le fromage râpé à la sauce. Verser le tout dans la lèchefrite du four huilée. Laisser refroidir complètement.

- Casser les œufs entiers dans un plat creux, ajouter un filet d'huile, saler et poivrer. Fouetter vivement. Verser la chapelure dans un autre plat creux. Faire chauffer un bain de friture dans une bassine. Découper la préparation au fromage en rectangles ou en carrés. Les passer d'abord dans les œufs battus, puis dans la chapelure. Les faire frire ensuite rapidement jusqu'à ce qu'ils soient bien dorés.

- Servir très chaud avec du Marin.

Trouvée sur `http://perso.wanadoo.fr/yves.huot-marchand/`

Chapitre 1

Contexte et motivations

Contexte

Grenoblois, j'aurais pu commencer cette introduction par une référence aux automates de Vaucanson (1709–1782) [Vau85]. En cette période de globalisation de l'économie [And96] et de la recherche de réduction des coûts, j'aurais aussi pu mettre un ou deux pointeurs vers des études sur l'impact socio-économique de la robotique et le désenchantement des industriels envers la robotique [DFE93]. Toutefois, je me contenterai de replacer l'asservissement visuel dans le contexte restreint de la robotique.

On distingue trois générations de robot [Gir97] d'après leur capacité à interagir avec le monde qui les entoure. Les robots de première génération, apparus au début des années 60, ne sont en fait que de simples automates programmables qui travaillent « en aveugle ». En effet, des automates ils conservent l'incapacité de réagir aux modifications de leur environnement. En revanche, par le biais de capteurs, dits proprioceptifs, leur état interne est connu, ce qui les rend programmables. Ils sont donc capables d'effectuer plusieurs tâches, contrairement aux simples automates. Ces tâches leur sont apprises, ou programmées, par enregistrement ou calcul d'une trajectoire qu'ils rejoueront au mieux. Cette capacité à être programmés pour une tâche, puis reprogrammés pour une autre en ont fait un outil très utilisé sur les chaînes de production.

Vint ensuite la deuxième génération de robots : en plus de leur capacité à connaître leur état interne, ces robots possèdent des capteurs additionnels, dits extéroceptifs, permettant de percevoir à des degrés divers l'état du monde. En particulier, l'état

d'avancement de la tâche qui leur est attribuée peut être évalué. Les capteurs les plus courants sont :

- les capteurs proximétriques (ultrasons, infrarouges) qui évaluent les distances dans un voisinage proche (de l'ordre du mètre) du robot en mesurant le temps écoulé entre l'envoi d'une onde et son retour ;

- les télémètres laser, basés sur le même principe mais avec une onde lumineuse, ont un champ d'action plus grand (de l'ordre de la dizaine de mètres) avec une plus grande précision ;

- enfin, les caméras vidéo fournissent une information dense, complexe et sans limite apparente de champ. Longtemps restreinte par la faible capacité de calcul des ordinateurs, la vision par ordinateur est désormais un domaine de recherche foisonnant.

Grâce à ces capteurs, on peut réaliser des robots aux tâches plus complexes. Ainsi, les robots mobiles, à roues, à pattes, voire même bipèdes, ont-ils pu voir le jour et des tâches complexes, telles que la soudure ou l'arrimage, ont-elles pu être automatisées.

Enfin, une troisième génération de robots est en train de naître. Ces robots possèdent des capacités déductives qui leur permettent, à partir des données capteurs et de description « haut niveau » de leur tâche d'en planifier la réalisation. Ces robots acquièrent, ou doivent acquérir, ainsi un certain degré d'autonomie de fonctionnement. L'exemple le plus médiatique en est *Sojourner*, envoyé en exploration sur Mars à l'été 1997, qui doit pouvoir fonctionner avec une interaction humaine restreinte, due à la distance Terre-Mars et la rotation de ces planètes.

Notre travail se trouve à l'intersection de la robotique et de la vision par ordinateur, puisque nous nous sommes intéressés au guidage visuel de robots. En particulier, nous nous sommes concentrés sur un type de guidage visuel : l'asservissement visuel. Cette discipline met en jeu un robot et une ou plusieurs caméras, ce qui nécessite de retrouver le lien de l'un à l'autre (étalonnage pince-caméra).

Notre travail relève aussi simultanément de la deuxième et de la troisième génération de robot. En effet, l'asservissement visuel correspond à la réalisation d'une tâche par utilisation d'informations visuelles et sans apprentissage d'une trajectoire. L'auto-étalonnage pince-caméra, quant à lui, est un effort d'augmenter l'autonomie du robot en relâchant les contraintes pratiques de l'étalonnage pince-caméra.

Asservissement visuel

La plupart des travaux en asservissement visuel se sont concentrés sur l'utilisation de points dans les images. Pourtant, dès 1990, Chaumette [Cha90] étudiait d'autres

primitives : droites, cercles ou sphères. Ces travaux n'ont depuis pas été souvent repris.

L'utilisation de points semble très restrictive pour bon nombre de problèmes industriels. L'une des raisons à cela tient du marquage de l'environnement, en général par des tâches blanches, pour simplifier le traitement d'images. Un constructeur automobile considèrera-t-il vraiment que moucheter de pastilles blanches toutes les pièces détachées d'un véhicule compense le gain de productivité engendré par la généricité de l'asservissement visuel ?

Certes, on peut considérer que ce problème de marquage sera résolu par l'évolution des techniques de traitement d'images et en particulier de suivi d'objets. Cependant, il reste un inconvénient majeur à l'utilisation de points : le fait même qu'il s'agisse de points. En effet, il ne suffit pas de détecter des points dans les images, encore faut-il qu'ils correspondent à des points physiques. Or, les points physiques n'existent pas forcément dans tout environnement. Prenons l'exemple du désassemblage de voitures usagées[GNTS96]. En théorie, il y a suffisamment de coins et recoins dans un compartiment moteur pour en extraire des points physiques. En pratique, cela se complique à cause des différents câbles traversant la scène, de la graisse déposée sur l'ensemble des pièces et gommant tout relief ou encore des reflets de lumière sur les flexibles ou sur les parties métalliques. On peut alors objecter que l'utilisation de points dans la commande n'implique pas forcément la détection de points dans les images. On peut ainsi construire un point comme étant l'intersection de deux segments concourants extraits des images. Toutefois, Chaumette a montré [Cha97] qu'il existe des minima locaux lorsque l'on utilise des points pour la commande.

Il faut donc envisager d'utiliser d'autres informations visuelles que des points. De récents travaux permettent ainsi de positionner une caméra par rapport à des jambons[Col99]. Sans aller jusqu'à ce cas extrême, nombreuses sont les applications où il existe des primitives géométriques dans la scène : inspection de conduites, suivi de route, assemblage complexe, etc.

Parmi ces applications réelles, on retrouve souvent des droites, car « l'*homo industrialis* » fabrique plus facilement des pièces polyèdriques que des objets courbes, esthétique mise à part. C'est pourquoi nous avons voulu ouvrir le champ d'application de l'asservissement visuel en approfondissant l'étude de l'utilisation des droites.

Enfin, nous nous restreindrons dans ce mémoire à utiliser une seule caméra, qui, de plus, est étalonnée. En effet, nous n'avons pas souhaité céder à la mode naissante de la stéréovision, car dans de nombreuses applications cette dernière n'est pas toujours utilisable. L'abandon de la stéréovision oblige alors à étalonner la caméra, même si cet étalonnage est autorisé à être approximatif.

La première partie de cette thèse, consacrée aux droites, se décompose de la manière suivante. Dans un premier temps, nous modéliserons le problème en reprenant les diverses représentations de droites et en définissant une tâche générique

de positionnement droite-sur-droite comme le suggère Hager pour le cas stéréoscopique[Hag97]. Puis, nous nous appliquerons à définir une loi de commande. Pour cela, nous passerons en revue un certain nombre de lois possibles, apparaîssant naturellement par mimétisme avec le cas des points, avant d'aboutir à notre loi finale. Cette loi a plusieurs caractéristiques :

1° Elle est analytique. Cela nous permet, contrairement aux commandes de la littérature, d'avoir des résultats de convergence globaux et d'exprimer analytiquement les singularités ;

2° Elle mélange des informations 2D et 3D ;

3° Elle est partiellement découplée en rotation et translation.

Afin de mettre en œuvre ces résultats, nous nous intéresserons à une application tirée d'une collaboration avec un chantier naval au travers du projet européen VIGOR [VIG01]. Elle permettra, en outre, de résoudre le problème de positionnement face à un trièdre orthogonal, qui est un cas dégénéré pour les algorithmes de vision.

Auto-étalonnage pince-caméra

Pour pouvoir commander un robot à l'aide d'une caméra qui lui est rigidement attachée, il est nécessaire de connaître la transformation euclidienne fixe qui les sépare : la transformation pince-caméra. Rappelons que sa détermination, l'étalonnage pince-caméra, se fait en général en comparant les mouvements du robot et ceux de la caméra.

Dans l'optique d'une simplification de l'utilisation de l'asservissement visuel et, partant, d'une augmentation de l'autonomie du robot, il est nécessaire de rendre la détermination de cette transformation plus aisée. À l'heure actuelle, il est nécessaire d'arrêter le système et de le placer face à une *mire de calibration*, c.-à-d. un objet de forme et taille connues avec une très grande précision. On peut donc envisager deux simplifications possibles : ne pas arrêter le système ou se passer d'une mire de calibration.

La première simplification revient à mettre à jour la transformation pince-caméra au fur et à mesure que le robot se déplace. Notre choix pour cela est d'utiliser les techniques de filtrage de Kalman pour définir une méthode d'étalonnage pince-caméra en ligne. La seconde simplification implique la définition d'une méthode d'auto-étalonnage pince-caméra.

Dans la deuxième partie de cette thèse, nous verrons qu'il est possible de réécrire le problème d'étalonnage pince-caméra sous forme purement linéaire (chapitre 1) et que cette formulation unifie la formulation des deux simplifications précédentes : étalonnage en ligne (chapitre 3) et auto-étalonnage (chapitre 2).

Chapitre 2

Notations

Dans cette thèse, nous utiliserons les notations suivantes :

- Les points 3D seront représentés par des majuscules en gras et italique : p. ex. $\boldsymbol{P}$;
- Les points 2D seront représentés par des minuscules en gras et italique : p. ex. $\boldsymbol{p}$;
- Les vecteurs seront représentés par des minuscules en gras : p. ex. $\mathbf{h}$;
- Les vecteurs unitaires seront de plus soulignés : p. ex. $\underline{\mathbf{u}}$;
- Nous ferons une exception à cette notation des vecteurs pour le torseur cinématique afin de garder sa notation traditionnelle : $\boldsymbol{\tau} = \left(\begin{smallmatrix}\mathbf{V}\\ \boldsymbol{\Omega}\end{smallmatrix}\right)$;
- Il nous arrivera de confondre un point 3D $\boldsymbol{P}$ avec le vecteur $\overrightarrow{\boldsymbol{OP}}$, où $\boldsymbol{O}$ est l'origine du repère ;
- Le produit vectoriel de deux vecteurs $\mathbf{v}_1$ sera noté $\mathbf{v}_1 \times \mathbf{v}_2$. La matrice antisymétrique qui lui est associée sera noté $As(\bullet)$: p. ex. $\mathbf{v}_1 \times \mathbf{v}_2 = As(\mathbf{v}_1)\mathbf{v}_2$;
- Nous préférerons noter le produit scalaire de deux vecteurs $\mathbf{v}_1$ et $\mathbf{v}_2$ par $\mathbf{v}_1^T\mathbf{v}_2$ plutôt que par la notation $\mathbf{v}_1 \cdot \mathbf{v}_2$ (plus cohérente avec la notation $\times$ du produit vectoriel). En effet, cela nous permettra de représenter de manière concise et agréable les expressions que nous rencontrerons. Par exemple, le produit mixte de trois vecteurs $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ est plus joli lorsqu'il est écrit comme ceci : $[\mathbf{a} \times \mathbf{b}]^T\mathbf{c}$ qu'avec les notations soit purement formelle : $((\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c})$, soit purement calculatoire : $(As(\mathbf{a})\mathbf{b})^T\mathbf{c}$. C'est d'ailleurs pour les mêmes raisons esthétiques que nous avons choisi la notation anglo-saxonne (pardon !) pour l'opérateur de transposition ;

- Les droites 3D (resp. 2D) seront représentées par une majuscule (resp. minuscule) droite entre parenthèses : p. ex. (D) (resp. (d)). Leurs coordonnées seront notées sans parenthèse : p. ex. D (resp. d) ;
- Enfin, les matrices seront notées en majuscules en gras : p. ex. $\mathbf{L}^T$.

Chapitre 3

État de l'art

Si la présente thèse contient deux thèmes, cet état de l'art se décompose, en revanche, en trois parties :

1° localisation tridimensionnelle ;

2° asservissement visuel ;

3° étalonnage pince-caméra.

En effet, nous verrons dans le reste de ce document que le problème de localisation tridimensionnelle apparaît de manière récurrente. Il nous a donc semblé nécessaire de rappeler ici quelques concepts clé de ce domaine qui nous seront utiles par la suite. En effet, que ce soit en asservissement visuel ou lors de l'étalonnage pince-caméra, il est (presque) toujours nécessaire de savoir répondre à l'une ou l'autre des deux questions suivantes : « Où se trouve la caméra vis-à-vis de la scène qu'elle observe ? » et « Quel est son déplacement ? »

3.1 Localisation tridimensionnelle

3.1.1 Pose ou reconstruction ?

Pour localiser une caméra dans l'espace par rapport à la scène qu'elle observe ou pour en connaître les déplacements, on dispose de deux types d'outils : calculs de pose ou reconstruction tridimensionnelle. L'utilisation de l'un ou de l'autre, ainsi que le résultat obtenu, dépendent des connaissances dont on dispose au sujet de la caméra et de la scène.

On appelle *pose* la position et l'orientation (c.-à-d. un élément de $SE(3)$) d'un objet par rapport au repère de la caméra (Figure 3.1). Formellement, le calcul de pose se fait à partir de correspondances entre points d'une image et points de l'espace

(correspondance 2D–3D). Cela impose donc d'avoir un modèle 3D de la scène que l'on observe. Éventuellement, le calcul de pose peut être complété par la détermination des paramètres intrinsèques de la caméra [FT87, LT88, Rob95]. On peut obtenir le déplacement de la caméra par composition de changements de repère entre 2 poses successives, en supposant la scène fixe. De manière duale, on peut obtenir le déplacement de l'objet par rapport à la caméra fixe (Figure 3.2).

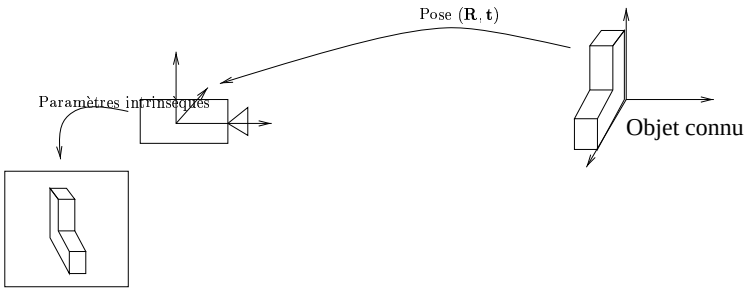


FIG. 3.1: *Calcul de pose : à partir d'un modèle de l'objet et de son image, on retrouve son orientation (*$\mathbf{R}$*) et sa position (*$\mathbf{t}$*) par rapport à la caméra.*

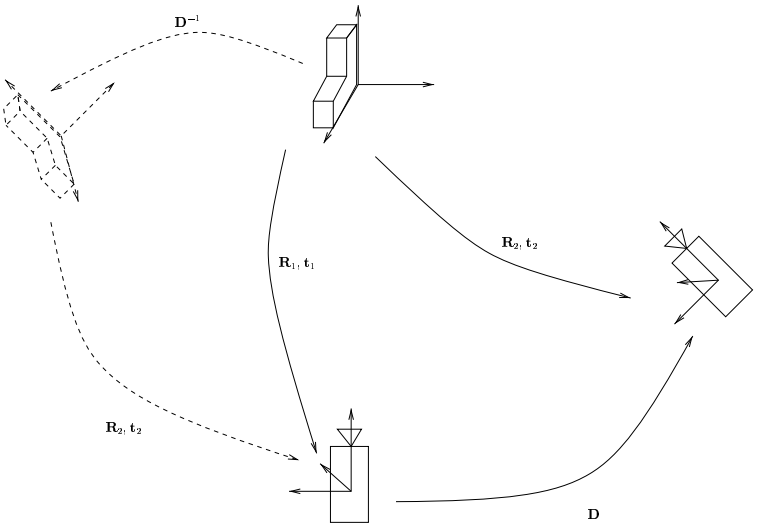


FIG. 3.2: *Dualité du mouvement de la caméra par rapport à un objet fixe et de celui de l'objet par rapport à la caméra fixe.*

Lors d'un calcul de reconstruction tridimensionnelle, on retrouve cette dualité. En effet, on parle aussi bien de reconstruction tridimensionnelle dans le cas d'une caméra mobile observant une scène fixe que dans celui-ci d'une caméra statique observant une scène mobile. Plaçons-nous dans le premier cas pour fixer les idées. Le principe de la reconstruction tridimensionnelle est de retrouver à partir de correspondances

de points d'une image à l'autre (correspondances 2D–2D) la position des points 3D qui ont généré les images. Il est donc inutile d'avoir un modèle de la scène. Si les paramètres intrinsèques sont inconnus, la reconstruction est projective ; sinon, elle est euclidienne. Dans ce dernier cas, la résolution effective des équations fournit les déplacements de la caméra entre les points de vue successifs (Figure 3.3).

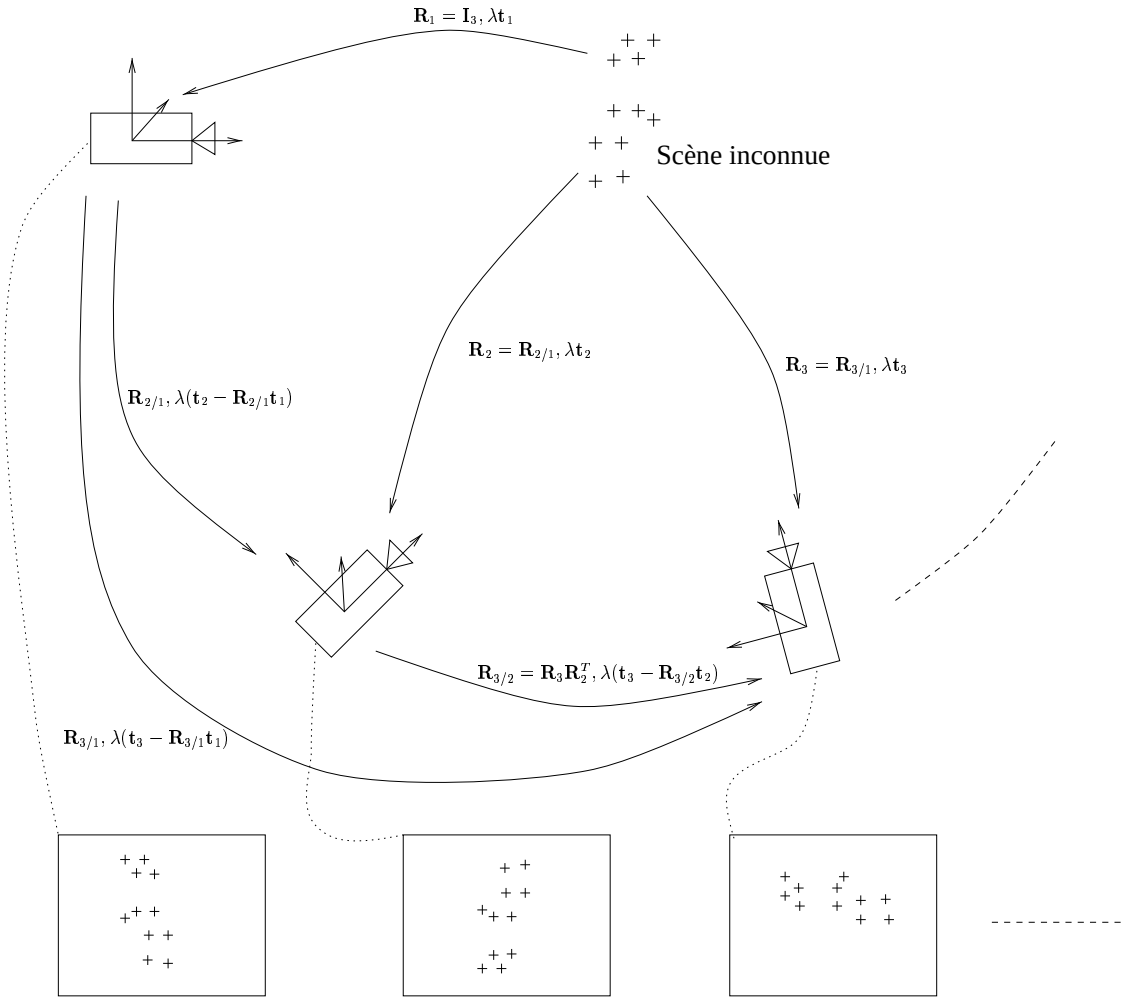


FIG. 3.3: *Reconstruction euclidienne.*

Néanmoins, comme la scène est inconnue, il reste une ambiguïté impossible à lever : "La caméra voit-elle une scène de grande taille depuis un point de vue éloigné ou est-elle proche d'une scène de petite taille ?". Lorsque l'on dispose d'un système stéréoscopique, c.-à-d. composé de plusieurs caméras, rigide, on peut lever cette inconnue si l'on dispose des positions relatives des caméras par étalonnage. Toutefois, dans la plupart des cas, l'ambiguïté de taille se traduit par l'apparition d'un facteur d'échelle inconnu dans les positions et les déplacements de la caméra.

3.1.2 À partir de points

Nous ne nous appesantirons pas sur le cas où le calcul de pose ou la reconstruction euclidienne se fait à partir de points extraits des images. Nous reportons plutôt le lecteur à la thèse de Christy [Chr98] pour une synthèse bibliographique.

3.1.3 À partir de droites

Calcul de pose

Dhome *et al.* [DRLR89] proposent une méthode pour le calcul de la pose d'un objet polyèdrique à partir de droites. L'orientation est obtenue par résolution, dans le cas général, d'un polynôme de degré 8. Cela peut donc donner lieu à l'existence de 8 solutions si le polynôme n'est pas identiquement nul. De plus, on notera que le degré élevé de ce polynôme peut engendrer des instabilités numériques. Dans un deuxième temps, un système linéaire fournit la position. Deux cas particuliers sont présentés qui se ramènent à la résolution d'un polynôme de degré 4 (il n'y a donc plus que 4 solutions au maximum si le polynôme n'est pas identiquement nul) pour l'orientation : droites coplanaires et droites concourantes. Dans ce dernier cas, d'ailleurs, le système linéaire utilisé pour déterminer la position n'est pas de rang plein. On doit alors se contenter de retrouver la droite sur laquelle se trouve le point d'intersection. En revanche, la distance à laquelle il se trouve sur cette droite ne peut être observée. Toutefois, si on peut voir un segment en entier, c.-à-d. que ses extrémités physiques sont bien détectées dans l'image, alors les auteurs proposent une formule pour trouver la distance inconnue. Enfin, on y rappelle quelques règles, tirées de [Low85], pour choisir la bonne orientation parmi celles issues des racines réelles des polynômes.

Quant à Chen [Che91a], il propose une méthode pour établir la pose à partir des correspondances entre 3 droites et 3 plans, dont une des applications est le calcul de la pose d'une caméra par rapport à un objet modélisé par des droites. En effet, une droite de l'espace se projetant en général en une droite de l'image, ces deux droites définissent un plan, appelé *plan d'interprétation*. Ce plan passant par l'origine, il est minimalement défini par la droite image. Cet article ne parle pas de l'obtention de la translation, mais propose une méthode polynômiale alternative à la précédente. Dans le cas général, le polynôme dont on cherche les racines est, ici aussi, de degré 8. Cette méthode exhibe de nombreux cas particuliers qui réduisent le degré du polynôme :

1° coplanarité de 3 droites ;

2° parallélisme de 2 des 3 droites (ou plans) ;

3° 3 plans mutuellement perpendiculaires : ce cas est dégénéré en vision puisque les 3 plans doivent intersecter le plan image ;

4° 3 plans s'intersectant le long d'une même droite : cela correspond au cas où les projections des 3 droites dans l'image sont concourantes (sans forcément l'être dans l'espace 3D) ;

5° 3 droites mutuellement perpendiculaires.

Application à un trièdre orthogonal Dans le Chapitre 3 de la partie consacrée à l'asservissement visuel de droites, nous positionnerons une caméra par rapport à un trièdre orthogonal. C'est pourquoi nous nous intéressons ici à ce cas particulier des calculs de pose. Les deux articles précités concluent à l'existence de 4 solutions possibles pour l'orientation en ce qui concerne 3 droites concourantes. De plus, on peut vérifier que la méthode de Dhome *et al.* se ramène à la résolution d'un polynôme *pair* de degré 4, lorsque les 3 droites sont perpendiculaires 2 à 2. Cela rejoint donc la conclusion de Chen à ce sujet. Il ne reste donc plus que 2 solutions possibles si le polynôme ne s'annule pas : une solution convexe et une solution concave.

Nous rappelons enfin que dans ce cas la profondeur est inobservable car la projection des trois droites est invariante dans l'image par translation le long de la droite de vue passant par leur intersection commune.

Autres méthodes Liu *et al.* [LHF90] proposent deux méthodes pour le calcul de la pose de la caméra. La première est non linéaire et nécessite 3 droites. La seconde est linéaire et en requiert 8 ; elle est basée sur une linéarisation du problème non-linéaire par une approximation au premier ordre de l'orientation relative de la caméra par rapport à l'objet. Elle n'est donc pas très robuste si l'on n'a pas d'estimation initiale suffisamment précise. Enfin, il est à peine fait mention de la non-unicité de la solution et on ne sait pas comment ils obtiennent la bonne.

Reconstruction

Il existe de nombreux travaux qui portent sur ce problème, de l'Euclidien [Fau93] au projectif [VLF95, Har94] en passant par l'affine [QK97] (que l'on consultera pour une bibliographie approfondie).

Comme les droites impliquent moins de contraintes que les points, il faut 3 vues (et non 2) et un nombre variable de droites dans chaque triplet d'images pour reconstruire : 6 droites pour un algorithme non-linéaire [LH86], 13 pour un algorithme linéaire [SA90] et 7 pour un calcul affine [QK97].

Ce besoin de trois images, même s'il n'est pas irrémédiable, n'est pas des plus simples à mettre en pratique dans notre cadre de travail. Plus grave, le nombre élevé de droites impose des contraintes sur la structure de la scène. Par conséquent, nous ne nous occuperons plus de reconstruction euclidienne à partir de droites dans ce document.

3.2 Asservissement visuel

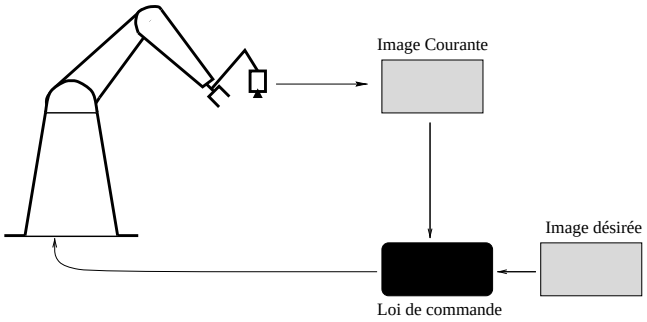


FIG. 3.4: *Le principe de l'asservissement visuel : déplacer le robot de manière à faire coïncider l'image courante avec l'image désirée.*

L'asservissement visuel consiste à déterminer une commande en boucle fermée sur retour visuel (Figure 3.4), le système de vision pouvant être formé d'une ou plusieurs caméras, statiques ou mobiles. Formellement, il s'exprime de la manière suivante :

1° la commande est l'entrée $\mathbf{u}$ d'un système produisant un mouvement euclidien relatif (dans $SE(3)$) entre le système de vision et une cible observée. L'ordre du système et la nature de l'espace d'entrée dépendent du type de robot utilisé, des hypothèses faite sur la dynamique de ce dernier et sur l'existence de boucles de commandes internes.

   L'entrée du système peut ainsi être : un torseur cinématique représentant les vitesses instantanées dans l'espace euclidien ($\tau \in TSE(3)$), les vitesses ou accélérations dans l'espace articulaire $\{\mathbf{q}\}$ ou encore les couples appliqués aux moteurs des articulations.

2° l'espace de sortie est celui où la tâche à réaliser (ou consigne) $\mathbf{y}^*$ est décrite et dans lequel la mesure courante $\mathbf{y}$ est obtenue à partir des données visuelles $\mathbf{s}$.

   La spécificité de l'asservissement visuel est que, quelles que soient les informations délivrées par le système de vision (objets géométriques dans l'image, estimation de la pose, etc.), une transformation projective est, explicitement ou non, présente dans la chaîne de traitement.

Le domaine étant désormais riche de centaines de références, nous ne classerons que les travaux qui nous semblent les plus significatifs. Le lecteur intéressé par une synthèse bibliographique approchant de plus près l'exhaustivité se référera en

premier lieu à celle produite par Corke [Cor93] ainsi qu'à l'article très pédagogique de Hutchinson *et al.* [HHC96].

Les études sur l'asservissement se sont réparties en trois classes : définition de l'espace de sortie, étude de ses variations et synthèse de lois de commande.

La définition de tâches visuelles fut l'un des premiers sujets traités et reste néanmoins un problème d'actualité [HDHM99]. En effet, on peut spécifier la tâche $\mathbf{y}$ de plusieurs façons à partir de $\mathbf{s}$ avec différentes propriétés. Nous verrons cela de plus près au paragraphe 3.2.1.

L'étude des variations de $\mathbf{y}$ s'est rapidement avérée cruciale. Si l'on note $\bar{\mathbf{r}}$ un élément de $SE(3)$ (une pose, donc) et en rappelant que $\mathbf{q}$ représente les variables articulaires, on a la formule générale :

$$\frac{\partial \mathbf{y}}{\partial \mathbf{q}} = \frac{\partial \mathbf{y}}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \bar{\mathbf{r}}} \frac{\partial \bar{\mathbf{r}}}{\partial \mathbf{q}} \tag{3.1}$$

La réponse à la question « Que connaissons-nous réellement dans cette expression ? » permet de résoudre le problème de la commande. Le troisième terme se réfère au système physique et inclut l'étalonnage du robot. S'il est suffisamment bien connu, on peut se contenter d'une commande définissant un torseur cinématique. Le premier terme est en principe bien connu puisqu'il dépend du choix de l'espace de sortie et de sa relation à l'espace des données visuelles. Le nœud du problème se situe donc dans l'expression $\frac{\partial \mathbf{s}}{\partial \bar{\mathbf{r}}}$ appelée Jacobien de l'image ou matrice d'interaction, et notée $\mathbf{L}$. Cette matrice dépend du type de primitives utilisées : points, droites, cercles, etc. Espiau *et al.* [ECR92] ont proposé une méthode générale de calcul de ces matrices d'interactions et ont montré [Esp93, Esp95] qu'elles dépendaient aussi, et en général explicitement, des paramètres d'étalonnage de la caméra, des mesures dans l'image et de la pose de la caméra par rapport à la scène. Enfin, « l'inversion » de cette expression permet de définir la commande, que l'on peut, par exemple, choisir du type proportionnelle. Cependant, il est, en pratique, quasi-impossible d'obtenir une valeur exacte de $\mathbf{L}$ et l'on est amené à en calculer une estimation $\hat{\mathbf{L}}$. Cette estimation peut être faite :

1° par identification en ligne, soit de la matrice d'interaction entière [JN95, HSA95] soit des paramètres qui la composent [KD94, HDE98] ;

2° en prenant $\hat{\mathbf{L}}$ égale à $\mathbf{L}^*$, valeur de $\mathbf{L}$ en position finale [ECR92, KRHK95] ou sur un maillage de l'espace de travail [DP96].

La troisième classe des travaux effectués part du constat qu'une commande proportionnelle ne compense pas forcément les erreurs statiques, telles que les frottements. Par ailleurs, les régimes transitoires sont affectés par la plus ou moins bonne connaissance de la matrice d'interaction. Ces problèmes de robustesse ont été traités par la branche automaticiennne du domaine : étude de sensibilité aux erreurs d'étalonnage [Esp93, MM91], commande optimale LQ [HEK96], entre autres.

En nous restreignant aux deux premières familles, nous pouvons affiner notre classement les travaux en matière d'asservissement visuel selon plusieurs critères non disjoints :

1° les primitives visuelles utilisées.

2° le nombre de caméras en fonction.

3° le degré d'étalonnage requis par le système.

Nous allons d'abord rappeler les méthodes les plus en vogue lorsque l'on extrait des points de l'image (§ 3.2.1). Puis, nous verrons ce qu'il en est pour d'autres primitives (§ 3.2.2). Dans un troisième temps, nous donnerons un aperçu des deux tendances principales actuelles : l'asservissement non étalonné (§ 3.2.3) et l'asservissement stéréoscopique (§ 3.2.4).

3.2.1 Asservissement visuel à partir de points

Dans un article fondateur, Weiss *et al.* [WSN87] ont défini deux critère de classification des lois de commandes visuelles :

1° Une commande peut se faire en position ou en vitesse.

Le premier mode de fonctionnement correspond à faire, au sein de la boucle de commande, des déplacements relatifs finis en boucle ouverte. Il n'est plus guère utilisé que pour le déboguage car il suppose un fonctionnement idéal du robot. Le second permet en effet de s'affranchir de l'incapacité des robots à réaliser un ordre de déplacement en boucle ouverte. On envoie donc un torseur cinématique.

Cela pose un autre type de problème : les images sont prises en mouvement. Cela limite donc la vitesse autorisée pour le robot. Ceci dit, les deux facteurs les plus limitants sont le temps de traitement des images (qui tend à se réduire) et le manque de prise en compte de la dynamique du robot à haute vitesse [CG96] ;

2° Une commande peut être effectuée dans l'image (commande 2D) ou dans l'espace euclidien (commande 3D).

La commande 2D [Cha90, ECR92] est ainsi appelée car elle travaille directement sur les données visuelles (c.-à-d. $\mathbf{y} = \mathbf{s}$), ici, les points. Elle est supposée être peu sensible aux erreurs de mesure, puisque l'on régule à 0 l'erreur sur le signal. La contrepartie de cette robustesse est qu'elle ne permet pas de spécifier explicitement la trajectoire dans l'espace du robot, ce qui peut créer des minima locaux ou générer des trajcetoires irréalisables (passage à l'infini)[Cha97].

La commande 3D peut être de deux types : commande dans l'espace des déplacements euclidiens ($SE(3)$) [WSN87, GNTS96, MDGD97, Ton97] ou

commande dans un espace de mesure 3D [HCM95, YW97], cette dernière ne se trouvant que dans le cas stéréo. La première, à l'inverse de la commande 2D, gère les trajectoires du robot puisqu'elle régule à 0 la différence entre la pose courante et la pose désirée. Ce type de commande requiert donc des calculs de pose ou de reconstruction, qui peuvent être sensibles aux erreurs de mesure, notamment en ce qui concerne la profondeur. Néanmoins, lorsque ces calculs sont suffisamment précis, la commande 3D et la commande 2D sont équivalentes [LG93]. Enfin, la commande 3D n'impose aucune trajectoire dans l'image, ce qui fait risquer de perdre la scène de vue.

Pour pallier ces problèmes, on peut définir des trajectoires de consigne (respectivement, $\mathbf{s}^*(t)$ ou $\mathbf{y}^*(t)$). On régule donc à 0 des erreurs plus faibles, limitant ainsi les déplacements dans l'espace ou dans l'image. On peut aussi tenter de choisir les primitives optimales pour l'application choisie, en 2D [FLM91] comme en 3D [JSW97].

Pour bénéficier des bonnes propriétés du 2D et du 3D tout en en limitant les inconvénients, la mode est désormais à la commande 2D1/2 [Mal98, MCB99]. Dans cette méthode, l'orientation est calculée en 3D alors que la translation dans les directions verticale et latérale est définie à partir d'un point de l'image. Pour ce qui est de la commande en profondeur et afin d'éviter de calculer une profondeur, ce qui est toujours peu précis, une méthode est proposée pour mesurer la profondeur relative entre la position courante et la position de consigne.

Dans la même optique d'éviter le calcul de la profondeur, Soatto et Perona [SP94] proposent un contrôleur basé sur l'espace de matrices essentielles [LH81].

3.2.2 Asservissement visuel à partir d'autres primitives

Quelques travaux [SBC94, GMOS95, MBG96, CC97, SVS97] commandent le flot optique des images, c.-à-d. la vitesse dans l'image et non la position. Ils requièrent donc l'utilisation de primitives additionnelles (points, temps avant impact) si l'on veut atteindre une position fixe (tâche de positionnement).

D'autres cherchent des informations pouvant caractériser l'image dans son intégralité : moments d'inertie ou transformée de Fourier [BJP93] ou analyse en composantes principales [DN96, NNM96] de l'image.

On trouve aussi des travaux de positionnement par rapport à des courbes planes [CAD95, CC96], dont les possibilités d'application sont, par nature, fortement restreintes. Une récente extension a été faite aux courbes non planes [Col99].

Certains encore vont chercher leur salut — le trouveront-ils? — dans les réseaux de neurones et les méthodes d'apprentissage [WVT96, SK93, WS97].

Enfin — redevenons sérieux ! — Chaumette *et al.* [Cha90, ECR92] ont déterminé des matrices d'interactions pour des primitives de type droite, cylindre, sphère ou ellipse. Ces travaux ont depuis été repris dans le cas des droites (p. ex. dans [Mot92, Deb96]). Les droites sont modélisées différemment par Hager [Hag97] dans le cas

stéréoscopique. Nous verrons dans la partie consacrée à l'asservissement visuel à partir de droites que ces deux approches n'ont pas forcément retenu la modélisation la plus simple.

3.2.3 Asservissement visuel non étalonné

On parle d'asservissement visuel non étalonné lorsque les différents paramètres intervenant dans la formation de l'image sont inconnus ou inutilisés.

Ainsi, Yoshimi et Allen [YA94, YA95] réalisent l'insertion d'une cheville dans un trou sans aucune connaissance des paramètres intrinsèques de la caméra ni de l'étalonnage pince-caméra (*cf.* § 3.3). Pour cela, ils fixent la cheville sur le dernier axe de rotation du robot et alignée avec lui ; de plus, une caméra est rigidement liée à ce même axe. Alors, on constate que le mouvement du centre du trou est une ellipse lorsque le robot tourne autour de son dernier axe et que cette ellipse est réduite à un point lorsque la cheville est correctement positionnée au-dessus du trou. Cette approche reste cependant assez restrictive puisqu'elle impose que la scène soit plane et que le mouvement du robot se fasse dans un plan parallèle à la scène avec une altitude constante. Similairement, Ghosh *et al.* [GTX+96] réussissent à saisir un objet sur un tourne-disque en observant le mouvement elliptique de l'objet dans l'image et le mouvement plan du robot dans l'espace à une altitude connue du tourne-disque.

On peut encore aller plus loin dans la suppression des connaissances *a priori* sur le système robot/caméra. En effet, Jägersand et Nelson [JN95] ainsi que Hosoda et Asada [HA94] « oublient » aussi bien l'étalonnage de la caméra que celui du robot et par un processus d'estimation en ligne retrouvent le Jacobien reliant les vitesses articulaires aux vitesses des points extraits des images (*Jacobien moteur-image*). C'est peut-être très utile lorsque l'on utilise un vieux robot avec du jeu dans les articulations (mais alors pourquoi ne pas aller jusqu'au niveau des courants moteurs?), cela semble exagéré dans la plupart des cas[1]. Il faut noter que cette estimation en ligne peut se détériorer par le fait de la convergence de la trajectoire vers une position finale. Il est alors indispensable d'utiliser des mouvements excitatoires, c.-à-d. surimposés à la trajectoire de contrôle, pour améliorer les résultats de l'estimation [SSV98]. Dans la même veine, Domingo et Pelechano [DP96] remplacent l'estimation en ligne du Jacobien moteur-image par son voisin le plus proche (au sens d'une mesure dans l'image) dans une grille de Jacobiens moteur-image pré-enregistrés, tirant parti de la continuité du lien moteur-image.

Sans aller aussi loin dans l'abandon des connaissances, on peut se contenter de simplifier le modèle de caméra en approximant le modèle projectif de la caméra par un modèle affine (voir [Chr98] pour une étude des différents modèles affines). Ainsi,

1. « On ne va pas utiliser un marteau-piqueur pour casser une noix » aurait dit mon prof. de maths en Taupe.

avec un modèle perspectif faible, on peut relier simplement les déformations affines de l'image au mouvement de la caméra [CAD95, CC96, SC96, HC94]. Cependant, pour ne pas engendrer de déformations perspectives, il faut se limiter à l'utilisation d'objets plans ou presque, c.-à-d. dont la profondeur est négligeable devant la distance qui les sépare de la caméra. De plus, l'approximation affine ne permet pas de distinguer les translations le long de l'axe vertical de l'image des rotations autour de l'axe horizontal, et vice-versa.

Enfin, Grosso *et al.* [GMOS95] utilisent un système stéréo non étalonné observant un bras manipulateur. Pour ce faire, ils estiment la rotation de l'outil terminal du bras par le calcul du flot optique dans les deux images. Les conditions de robustesse de leur loi de commande sont si faibles qu'elles leur permettent d'éviter tout étalonnage. En revanche, ces travaux ne semblent pas mettre en jeu de commande de l'orientation de l'outil, seule sa position importe.

En conclusion, lorsque l'on essaie de se passer d'étalonnage, il faut s'imposer de nouvelles contraintes : sur les mouvements autorisés, sur les scènes observées ou encore l'utilisation de lourdes techniques numériques. De plus, à notre connaissance, aucun résultat de stabilité des lois de commande non étalonnées n'existe à ce jour.

3.2.4 Asservissement visuel stéréo

L'asservissement visuel stéréoscopique rejoint très rapidement l'asservissement visuel non étalonné puisque l'on retrouve dans cette catégorie certains des travaux vus précédemment [GMOS95, HC94, JN95]. Deux raisons semblent expliquer ce rapprochement : la deuxième caméra apporte des informations supplémentaires permettant de supprimer les contraintes dues au manque d'étalonnage ; de plus, la stéréoscopie, même sans étalonnage, apporte une information 3D qui n'est disponible dans le cas monoculaire que via les paramètres intrinsèques.

Pourtant l'asservissement visuel stéréo a d'abord été étalonné. Par exemple, Allen *et al.* [ATYM93, ATYM94] pouvaient grâce à deux caméras étalonnées indépendamment se saisir d'un train électrique. Cependant, la saisie était faite en boucle ouverte après prédiction de la position du train à l'instant de la saisie. Similairement, Joshi et Sanderson [JS96] placent un fil dans son ampoule par utilisation alternative de deux asservissements visuels monoculaires étalonnés. Enfin, Maru *et al.* [MKNM93] (et semblablement Hager *et al* [HCM95, Hag97]) utilisent simultanément deux caméras étalonnées pour faire une reconstruction euclidienne de l'objet observé, en déduisent un Jacobien de type asservissement visuel 2D pour chaque image et les empilent pour former un Jacobien stéréo.

Pour revenir au cas non étalonné, passons par la solution « miracle » de cette fin de siècle : le réseau de neurones. Pagel *et al.* [PMvdM98] y ont recours pour garder un objet au centre de l'image (tâche de fixation).

Deux approches [GMOS95, Xie97] proposent des solutions ne considérant qu'un

seul point : l'extrémité de l'outil terminal observé par deux caméras non étalonnées. Par conséquent, la commande ne se fait qu'en translation.

Sur la base de [HA94], Hosoda *et al.* [HSA95] proposent une méthode simple pour éviter les obstacles grâce à la stéréo. Cependant, cette méthode n'en est encore qu'au stade de la planification et du suivi de la trajectoire ainsi définie. Il manque encore une étude plus fine des trajectoires générées et une mise à jour en temps réel de ces dernières durant la boucle d'asservissement.

Plus récemment, des travaux ont débuté pour modéliser plus proprement l'asservissement visuel stéréo non étalonné en faisant appel à la géométrie projective [HD97, HDHM99, RH99].

3.3 Étalonnage pince-caméra

Lorsque l'on veut commander un robot en commande référencée capteur, il est nécessaire de pouvoir relier les informations en provenance du capteur au repère de commande effectif du robot. En particulier, dans le cadre de l'asservissement visuel, on veut connaître la position/orientation (ou *transformation rigide*) qui relie le repère de commande du robot au repère dans lequel sont effectuées les opérations de vision (calcul de pose, reconstruction, etc.). Ce problème est traditionnellement appelé *étalonnage pince-caméra* (en anglais, *hand-eye calibration*[2]).

Il existe sous deux formes duales [DH98] :

1° une ou plusieurs caméra est montée sur un robot. On a alors besoin de connaître la transformation rigide entre le repère de la caméra et le repère de commande du robot[3] ;

2° une ou plusieurs caméras observent un objet attaché rigidement à l'outil terminal du robot. On a alors besoin de déterminer la transformation rigide entre le repère de l'objet et le repère de commande du robot.

Nous nous plaçons ici dans le cas d'un système composé d'un bras robotique sur lequel est fixée une caméra. Nous appellerons repère pince, le repère de commande du robot et, tout naturellement, repère caméra, le repère associé à la caméra.

3.3.1 Transformation pince-caméra et asservissement visuel

Afin de mettre en évidence la nécessité de connaître la transformation pince-caméra lors de l'asservissement visuel, prenons comme exemple simple, l'asservissement en position avec convergence exponentielle par *fonction de tâche*. Amplement

2. On trouve parfois ce terme pour désigner le calcul de la pose d'une caméra par rapport à une scène [LMH96, Hor86, NTG92, vALV94]

3. La constance de ce lien est utilisée par Dias *et al.* [DdAAB91] pour optimiser les paramètres intrinsèques d'une caméra déplacée par un robot.

développée dans [SLBE91], la notion de fonction de tâche est présentée dans [Cha90] et [ECR92] avec son application à l'asservissement visuel.

Dans cet exemple, nous avons :

$$\left\{ \begin{array}{rcl} \tau & = & -\lambda \mathbf{e} \\ \mathbf{e} & = & \mathbf{L^T}^{+}(\mathbf{s} - \mathbf{s}^*) \end{array} \right.$$

où τ est le torseur cinématique appliqué au robot et exprimé dans le repère pince; λ est un facteur de gain; $\mathbf{e}$ est la fonction de tâche que l'on veut réguler à 0; $\mathbf{L^T}$ est appelée matrice d'interaction et est exprimée dans le repère pince; enfin, $\mathbf{s}$ (resp. $\mathbf{s}^*$) représente la primitive image courante (resp. désirée).

On montre alors que la matrice d'interaction $\mathbf{L^T}$ est fonction des paramètres intrinsèques de la caméra (α_u et α_v, les facteurs d'échelle des axes de l'image et θ l'angle entre ces axes), d'une matrice d'interaction canonique exprimée dans le repère caméra ($\mathbf{L_C^T}$) et de la transformation pince-caméra ($\mathbf{X} = (\mathbf{R_X}, \mathbf{t_x})$) :

$$\mathbf{L^T} = \begin{pmatrix} -\alpha_u & \alpha_u cotg\theta \\ 0 & -\frac{\alpha_v}{sin\theta} \end{pmatrix} \cdot \mathbf{L_C^T} \cdot \begin{pmatrix} \mathbf{R_X} & -\mathbf{R_X} As(-\mathbf{R_X}^T \mathbf{t_x}) \\ 0 & \mathbf{R_X} \end{pmatrix}$$

où $As(\bullet)$ est la matrice antisymétrique associée au produit vectoriel et $\mathbf{L_C^T}$ dépend de la position de la caméra et du type de primitive image utilisé (points, droites, etc.).

3.3.2 Transformation pince-caméra et résolution

Pour faire apparaître la transformation pince-caméra, on peut considérer le déplacement d'un système constitué d'une pince et d'une caméra fixées l'une à l'autre (Figure 3.5)

Lorsque la pince se déplace selon la transformation rigide $\mathbf{B} = (\mathbf{R}_B, \mathbf{t}_B)$, la caméra se déplace, elle, selon la transformation rigide $\mathbf{A} = (\mathbf{R}_A, \mathbf{t}_A)$. $\mathbf{A}$ et $\mathbf{B}$ sont liées par [SA89] :

$$\mathbf{AX} = \mathbf{XB}$$

C'est cette équation matricielle qu'il s'agit de résoudre après avoir obtenu $\mathbf{B}$ grâce au modèle cinématique du robot et $\mathbf{A}$ par des calculs de pose (en général) ou par reconstruction euclidienne (dans cette thèse).

3.3.3 Méthodes de référence

Sont classées dans cette catégorie, les méthodes qui nous serviront pour l'évaluation expérimentale des nôtres. Nous mentionnerons dans ce même paragraphe les approches voisines.

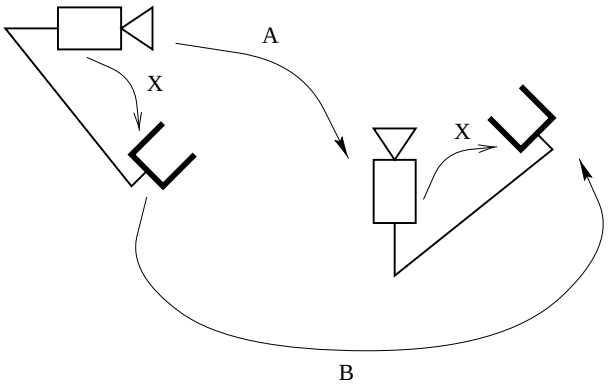


FIG. 3.5: *Les deux positions d'un même système pince-caméra séparés par le déplacement* **A** *de la caméra, auquel est associé le déplacement* **B** *du robot au moyen de la transformation pince-caméra* **X**.

Axe/angle

Cette méthode, présentée dans [TL89], résout le problème d'étalonnage pince-caméra en 2 étapes. On détermine d'abord l'axe $\underline{\mathbf{n}}$ et l'angle θ de la rotation pince-caméra à l'aide des équations suivantes :

$$As(\underline{\mathbf{n}}_A + \underline{\mathbf{n}}_B)\underline{\mathbf{n}} = \underline{\mathbf{n}}_A - \underline{\mathbf{n}}_B \quad (3.2)$$

$$\theta = 2\arctan(\|\underline{\mathbf{n}}\|) \quad (3.3)$$

où $\underline{\mathbf{n}}_A$ (resp. $\underline{\mathbf{n}}_B$) est l'axe de rotation du mouvement effectué par la caméra (resp. par le robot). Puis, l'on construit la matrice de rotation $\mathbf{R}$ associée à $\underline{\mathbf{n}}$ et θ afin de trouver la translation pince-caméra $\mathbf{t}$ par la méthode des moindres carrés appliquée au système :

$$(\mathbf{R}_B - \mathbf{I})\mathbf{t} = \mathbf{R}\mathbf{t}_A - \mathbf{t}_B$$

Cette méthode présente l'avantage d'être simple à mettre en œuvre mais elle repose essentiellement sur l'hypothèse que les mouvements de rotation sont suffisamment amples pour pouvoir en extraire les axes. De plus, l'axe $\underline{\mathbf{n}}$ est obtenu par résolution aux moindres carrés, ce qui rend les résultats sensibles aux mesures aberrantes.

Des méthodes similaires sont proposées avec des variations sur le thème de l'estimation de la rotation pince-caméra. Shiu et Ahmad [SA89] trouvent la rotation en calculant une solution particulière pour chaque mouvement. La solution générale pour chaque mouvement dépend alors d'un seul angle. La rotation pince-caméra est donc la rotation qui est solution pour tous les mouvements et est obtenue en trouvant les angles permettant d'unifier toutes ces solutions particulières. Park et

Martin [PM94], quant à eux, se basent sur les groupes de Lie pour trouver la rotation par moindres carrés linéaires. Enfin, dans le but d'étalonner une tête stéréo articulée, Li et Betsis [LB95] trouvent la rotation pince-caméra en ramenant les rotations de la pince et celles de la caméra à des rotations canoniques de la forme $\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Quaternions duaux unitaires

Cette méthode, présentée dans [DBC96], est basée sur la représentation des déplacements rigides par des quaternions duaux unitaires, une extension des quaternions, de la forme :

$$(\mathbf{R}, \mathbf{t}) \longleftrightarrow \mathcal{Q} \triangleq \begin{pmatrix} \mathfrak{q}^T \\ \mathfrak{q}'^T \end{pmatrix} \triangleq \begin{pmatrix} q_0 & \mathbf{q}^T \\ q'_0 & \mathbf{q}'^T \end{pmatrix} \text{ avec } \|\mathcal{Q}\| = 1$$

où $\mathfrak{q}$ et $\mathfrak{q}'$ sont des quaternions unitaires. Le premier est celui utilisé communément pour représenter les rotations. Le second contient une représentation de la rotation. Quant à q_0 et $\mathbf{q}$, ce sont les parties scalaire et vectorielle de $\mathfrak{q}$. Avec cette représentation, l'équation $\mathbf{AX} = \mathbf{XB}$ s'écrit :

$$\mathcal{A} = \mathcal{Q}\mathcal{B}\bar{\mathcal{Q}} \tag{3.4}$$

où $\mathcal{A}$ est associé à la transformation rigide $\mathbf{A}$, $\mathcal{B}$ à $\mathbf{B}$ et $\mathcal{Q}$ à $\mathbf{X}$, et mène à :

$$\begin{pmatrix} \mathbf{a}-\mathbf{b} & As(\mathbf{a}+\mathbf{b}) & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ \mathbf{a}'-\mathbf{b}' & As(\mathbf{a}'+\mathbf{b}') & \mathbf{a}-\mathbf{b} & As(\mathbf{a}+\mathbf{b}) \end{pmatrix} \begin{pmatrix} \mathfrak{q} \\ \mathfrak{q}' \end{pmatrix} = 0 \tag{3.5}$$

En empilant $n \geq 2$ matrices de ce type, correspondant à n mouvements du système pince-caméra, on obtient une matrice $\mathbf{T}$ de dimension $6n \times 8$ et de rang 6 (en l'absence de bruit). Alors, la solution du système est une combinaison linéaire de 2 vecteurs indépendants du noyau. Ces vecteurs sont obtenus par SVD [PTVF92]. Ils sont, dans ce cas, les 2 derniers vecteurs singuliers droits (notés $\mathbf{v}_7$ et $\mathbf{v}_8$) correspondant aux 2 plus petites valeurs singulières de $\mathbf{T}$:

$$\begin{pmatrix} \mathfrak{q} \\ \mathfrak{q}' \end{pmatrix} = \lambda\mathbf{v}_7 + \mu\mathbf{v}_8$$

Comme le résultat doit être un quaternion dual unitaire, nous obtenons deux équations en λ et μ dont la solution détermine la transformation pince-caméra :

$$\lambda^2\mathbf{u}_1^T\mathbf{u}_1 + 2\lambda\mu\mathbf{u}_1^T\mathbf{u}_2 + \mu^2\mathbf{u}_2^T\mathbf{u}_2 = 1 \tag{3.6}$$

$$\lambda^2\mathbf{u}_1^T\mathbf{v}_1 + \lambda\mu\left(\mathbf{u}_1^T\mathbf{v}_2 + \mathbf{u}_2^T\mathbf{v}_1\right) + \mu^2\mathbf{u}_2^T\mathbf{v}_2 = 0 \tag{3.7}$$

avec les conventions $\mathbf{v}_7 = (\mathbf{u}_1^T, \mathbf{v}_1^T)^T$ et $\mathbf{v}_8 = (\mathbf{u}_2^T, \mathbf{v}_2^T)^T$.

Cette méthode permet le calcul simultané de la rotation et de la translation pince-caméra. Cependant, elle repose elle aussi sur l'hypothèse que les rotations sont conséquentes. Enfin, l'application des contraintes pour trouver λ et μ nécessite de trouver une solution réelle à une équation polynômiale du second degré dont il n'est pas prouvé que le déterminant soit positif.

Partant d'une formulation équivalente, Kim [Kim96a, Kim96b] propose une solution analytique pour les mesures parfaites et se rabat sur une résolution itérative par descente de gradient lorsque les mesures sont erronées.

Si l'utilisation des quaternions duaux est explicite dans ces deux méthodes, elle est moins visible dans les travaux de Chou *et al.* [CK91, YCC95]. Pourtant, ceux-ci résolvent le problème en représentant aussi bien la rotation que la translation pince-caméra à l'aide de quaternions.

Minimisation non linéaire

Cette méthode, présentée dans [HD95], réalise une minimisation non linéaire, selon l'algorithme de Levenberg-Marquardt [PTVF92], de la fonction de coût suivante qui utilise la représentation des rotations par des quaternions unitaires :

$$\begin{aligned} f(\mathfrak{q},\mathbf{t}) \;=\; & \lambda_1 \Sigma_{i=1}^{n} \|\underline{\mathbf{n}}_{A_i} - \mathfrak{q} * \underline{\mathbf{n}}_{B_i} * \bar{\mathfrak{q}}\|^2 \\ & + \lambda_2 \Sigma_{i=1}^{n} \|\mathfrak{q} * \mathbf{t}_{B_i} * \bar{\mathfrak{q}} - (\mathbf{R}_{A_i} - \mathbf{I})\mathbf{t} - \mathbf{t}_{A_i}\|^2 \\ & + \lambda(1 - \mathfrak{q}^T\mathfrak{q})^2 \end{aligned}$$

où n est le nombre de mouvements, λ est un multiplicateur de Lagrange, λ_1 et λ_2 sont deux poids au choix de l'utilisateur et les notations restantes sont identiques à celles des méthodes précédentes.

Cette méthode non-linéaire nécessite une initialisation proche de la solution sans quoi la convergence est longue. De plus, elle nécessite aussi des rotations de grande amplitude, ainsi que le choix de poids par l'utilisateur. Elle permet cependant une grande précision de résolution et a une extension simple dans le cas où l'on monte une paire stéréo sur le robot [HDBM94]

3.3.4 Autres méthodes

Les méthodes suivantes ne présentent pas assez de similarités avec celles que nous avons choisies comme références pour être classées précédemment.

Zhuang s'est fait le champion des commentaires d'articles concurrents [ZR91, ZR92, Zhu97] et de la résolution itérative du problème d'étalonnage pince-caméra par la méthode de Gauss-Newton [ZS93, ZQ94, ZWR95, Zhu98a]. Pour cela, il définit un « *identification Jacobian* » à partir de l'équation $\mathbf{AX} = \mathbf{XB}$. Lee et Ro [LR96]

proposent une méthode d'auto-étalonnage simultané d'un robot et du lien pince-caméra, similaire à [ZWR95] à l'exception qu'ils représentent les rotations par des quaternions.

Une méthode beaucoup plus originale est proposée par Rémy [RDLD97, Rém98]. Elle permet, à partir de l'observation d'une mire quelque peu particulière, d'obtenir simultanément la transformation pince-caméra et la position de la mire dans le repère de base du robot. À l'inverse des résultats semblables obtenus par [ZRS94, DH98] en résolvant une équation homogène de la forme $\mathbf{AX} = \mathbf{YB}$, cette méthode ne nécessite pas de calcul du déplacement de la caméra. En effet, elle minimise l'erreur de reprojection des points de la mire en fonction des deux transformations inconnues. Toutefois, pour obtenir l'information nécessaire à la résolution du système, il faut tout de même déplacer la caméra.

Il faut encore citer Chen [Che91b] pour son étude géométrique, basée sur la théorie des torseurs, qui fournit des conditions d'existence et d'unicité de la solution.

Enfin, Wei *et al.* [WAH98] proposent une méthode d'auto-étalonnage caméra et pince-caméra en observant un simple point, mais au prix de lourds calculs d'identification et d'une procédure d'étalonnage restreignant l'espace des mouvements possibles du couple caméra/robot.

3.3.5 Limitations

Les principales limitations que l'on peut observer dans toutes ces méthodes sont de deux types :

1° l'utilisation de représentations réduites (axe/angle, quaternions) des rotations qui sont mal définies pour une rotation nulle ;

2° la mise en œuvre de techniques lourdes d'optimisation numérique.

Nous nous proposons donc dans la deuxième partie de cette thèse de formuler une méthode purement linéaire basée sur la représentation redondante mais unique et toujours bien définie des rotations par des matrices orthonormales. Cette méthode a, de plus, l'avantage de servir de base à une méthode d'auto-étalonnage et une méthode d'étalonnage en ligne.

Asservissement visuel à partir de droites

Féra à la Lugrinoise

Pour 8 personnes
Ingrédients : 2 kg de féra fraîchement pêchée dans le Léman, 8 carottes, 8 blancs de poireaux, 1 verre de vin blanc, un petit pot de crème fraîche, 2 citrons, beurre, sel, poivre

- Passer les carottes au moulin à julienne. Les mettre à suer dans un morceau de beurre pendant 4 à 5 mn avec du sel et un soupçon de poivre.
- Couper les blancs de poireaux en tronçons les plus minces possible. Les mettre à suer dans un rien de beurre. Surtout qu'ils ne rissolent pas ! Ajouter encore un peu de sel et de poivre.
- Mettre les deux lits de légumes dans un grand plat allant au four. Y déposer la féra salée et poivrée à l'intérieur. Arroser des jus des citrons et du verre de vin blanc. Napper la féra de crème.
- Mettre à four chaud 30 à 35 mn.
- Servir chaud, accompagné de Féternes bien frais.

Chapitre 1

Modélisation

1.1 Introduction

Avant de pouvoir définir des lois de commandes pour pouvoir asservir visuellement un robot face à une scène constituée de droites, il faut se poser quelques questions.

1. Comment représente-t-on des droites?

   Il faut se poser cette question à la fois pour les droites de l'espace et pour les droites extraites des images. En effet, contrairement au cas des points, il existe plusieurs représentations possibles pour une droite.

2. Comment exprime-t-on le mouvement apparent des droites?

   Le mouvement apparent d'une droite 3D est le mouvement perçu dans l'image de la projection de cette droite. Il dépend, entre autres, du mouvement de la caméra et est exprimé différemment selon la représentation utilisée pour les droites.

3. Que signifie superposer deux droites dans l'image?

   En asservissement visuel 2D, on cherche à superposer un ensemble de points dans l'image sur une position désirée. Dans cette optique, on peut vouloir « superposer » un ensemble de droites dans l'image sur une position désirée. Superposer signifie annuler la distance d'un objet à un autre ou à un ensemble d'objets. Or, si la distance de deux points est évidemment leur distance euclidienne, il n'est pas aussi naturel de définir une métrique pour les droites.

1.2 Représentation des droites

Dans ce paragraphe, nous allons passer en revue les représentations de droites du plan et de l'espace les plus courantes. Puis, nous verrons comment elles ont été utilisées en asservissement visuel. Enfin, nous définirons la représentation des droites, basée sur la notion de coordonnées de Plücker binormées, qui servira de base à la suite de ce travail.

1.2.1 Représentation 2D

Il existe quatre manières principales de représenter les droites du plan. Elles sont toutes basées sur l'équation classique d'une droite (d) :

$$(\mathsf{d}) : ax + by + c = 0$$

où les coefficients (a, b, c) sont connus et (x, y) sont les coordonnées d'un point du plan, qui appartient à (d) si et seulement si ses coordonnées vérifient l'équation précédente.

La première représentation de la droite (d) consiste tout simplement à l'identifier au vecteur contenant les trois coefficients de son équation :

$$\mathsf{d} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{1.1}$$

On peut alors exprimer l'équation de (d) sous la forme :

$$(\mathsf{d}) : \boldsymbol{p}^T \mathsf{d} = \boldsymbol{p}^T \begin{pmatrix} a \\ b \\ c \end{pmatrix} = 0 \tag{1.2}$$

où $\boldsymbol{p}$ est un point de l'image de coordonnées homogènes $(x, y, 1)^T$.

Cette représentation n'est pas unique car ce triplet est défini à un facteur multiplicatif près. Deux normalisations peuvent être utilisées qui définissent les deux représentations suivantes. La première impose que la norme euclidienne du triplet soit unitaire et définit la deuxième représentation des droites 2D :

$$\mathsf{d} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \text{ avec } a^2 + b^2 + c^2 = 1 \tag{1.3}$$

On s'est ainsi ramené à une seule ambiguïté de signe (donc d'orientation pour la droite) en normalisant ce triplet. Notons que cette ambiguïté est intrinsèque à toute

représentation d'une droite et qu'elle se retrouvera donc dans toutes les représentations suivantes.

La seconde façon de normaliser le triplet consiste à le diviser par la norme euclidienne de ses deux premiers coefficients. Cela correspond donc à la définition suivante :

$$\mathsf{d} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \text{ avec } a^2 + b^2 = 1$$

Du fait de cette contrainte, on peut réécrire cette définition sous la forme suivante et obtenir la troisième représentation :

$$\mathsf{d} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ -\rho \end{pmatrix} \quad \text{ou bien} \quad \mathsf{d} = \frac{1}{\sqrt{a^2+b^2}} \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{1.4}$$

On peut alors s'inspirer de cette troisième représentation de (d) pour définir la quatrième, cette fois-ci minimale :

$$\mathsf{d} = \begin{pmatrix} \rho \\ \theta \end{pmatrix} \tag{1.5}$$

dans laquelle, θ et ρ sont tels que :

$$(\mathsf{d}) : x\cos\theta + y\sin\theta - \rho = 0$$

On notera que cette représentation est minimale mais pas tout à fait unique car $(\rho, \theta + 2k\pi)$ et $(-\rho, \theta + (2k+1)\pi)$ représentent la même droite.

1.2.2 Représentation 3D

Nous considérerons ici trois manières de représenter une droite (D) dans l'espace. La première est algébrique puisqu'elle considère une droite comme l'intersection de 2 plans :

$$(\mathsf{D}) : \begin{cases} a_1 x + b_1 y + c_1 z + d_1 = 0 \\ a_2 x + b_2 y + c_2 z + d_2 = 0 \end{cases} \tag{1.6}$$

Comme seuls quatre paramètres suffisent à définir une droite de l'espace, on peut lier ces deux équations par des contraintes additionnelles [Bou93].

On en rencontre [Fau93, NFV93, Plü65] une variante minimale, parfois appelée représentation (a, b, p, q), qui est telle que :

$$(\mathsf{D}) : \begin{cases} x = az + p \\ y = bz + q \end{cases}$$

Asservissement visuel à partir de droites

Elle représente les droites qui ne sont ni perpendiculaires à l'axe des z, ni parallèles au plan xy. Dans le cas de la vision, ce sont les droites qui ne sont ni perpendiculaires à l'axe optique, ni parallèles au plan image. Pour prendre en compte toutes les droites de l'espace, il faut alors effectuer des permutations circulaires sur x, y et z. On obtient alors en fait trois représentations similaires, non disjointes et qui, ensemble, regroupent toutes les droites de l'espace. Nous ne poursuivrons pas plus l'évocation de cette représentation dont la discontinuité ne pourra qu'être gênante dès lors que nous ferons de la commande.

La seconde représentation, plus légère, consiste en un couple point-direction qui exprime que par un point donné ne passe qu'une seule droite selon une direction donnée :

$$\mathsf{D} = \begin{pmatrix} \boldsymbol{P} \\ \underline{\mathbf{u}} \end{pmatrix} \tag{1.7}$$

où $\boldsymbol{P}$ est un point 3D quelconque de (D) et $\underline{\mathbf{u}}$ est un vecteur directeur unitaire de (D). On remarquera que cette représentation n'est pas unique car n'importe quel point de D peut être choisi comme point de référence.

Coordonnées de Plücker — La troisième représentation est une variante euclidienne des coordonnées de Plücker [Plü65], qui, elles, relèvent de la géométrie projective [Fau93, SK52]. C'est une représentation suffisamment intéressante pour que nous lui dédiions le reste de ce paragraphe. On trouvera en Annexe A une présentation des coordonnées de Plücker projectives et comment on obtient de ces dernières les coordonnées euclidiennes.

La différence entre les coordonnées de Plücker projectives et euclidiennes vient du fait que l'espace euclidien est muni d'un produit scalaire et donc d'une norme. De ce fait, les coordonnées projectives, qui sont définies à un facteur d'échelle près, peuvent être normées pour obtenir les coordonnées euclidiennes. On trouve ainsi parfois l'appellation *coordonnées de Plücker normées* [PPR98] pour désigner les coordonnées de Plücker euclidiennes.

Comme nous resterons, par la suite, dans le monde euclidien, nous nous autoriserons l'abus de langage consistant à désigner par « coordonnées de Plücker » ces coordonnées euclidiennes de Plücker. Dans le monde euclidien, donc, les coordonnées de Plücker de (D) sont définies comme :

$$\mathsf{D} = \begin{pmatrix} \mathbf{h} \\ \underline{\mathbf{u}} \end{pmatrix} \tag{1.8}$$

Le vecteur unitaire $\underline{\mathbf{u}}$ désigne toujours un vecteur directeur de (D) et $\mathbf{h}$ est le vecteur défini par le produit vectoriel :

$$\mathbf{h} = \boldsymbol{P} \times \underline{\mathbf{u}} \tag{1.9}$$

où $\boldsymbol{P}$ est, ici encore, un point quelconque de (D). On remarquera que $\mathbf{h}$ est indépendant du choix de $\boldsymbol{P}$. Par conséquent, cette représentation est unique au choix de l'orientation de la droite près.

On notera que lorsque $\mathbf{h}$ est nul, alors (D) passe par l'origine et réciproquement. Il s'agit du cas dégénéré où la droite (D) se projette en un point dans l'image. En pratique, cela signifie que la droite n'est pas visible. Dans le contexte de l'asservissement visuel, une exception d'erreur de détection sera alors levée, avant même qu'un traitement mathématique puisse être effectué pour la commande. Ainsi, en pratique, ce cas dégénéré n'interviendra jamais et, par conséquent, nous pouvons conserver pour la suite cette hypothèse non restrictive.

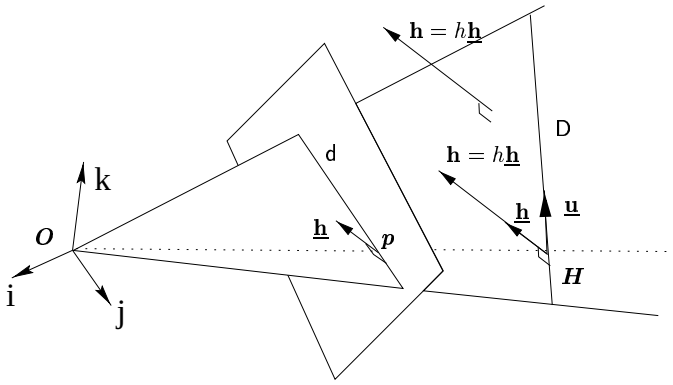


FIG. 1.1: *Interprétation géométrique des coordonnées de Plücker d'une droite.*

D'habitude, les coordonnées de Plücker s'interprètent géométriquement (voir Figure 1.1) de la manière suivante : la droite (D) est la droite de vecteur directeur unitaire $\underline{\mathbf{u}}$ et passant par le point $\boldsymbol{H} = \underline{\mathbf{u}} \times \mathbf{h}$, qui est le point de (D) le plus proche de l'origine [NFV93]. On remarque aussi que le vecteur $\mathbf{h}$ est perpendiculaire au plan, appelé *plan d'interprétation*, passant par l'origine et contenant (D) ainsi que la projection (d) de cette dernière dans l'image.

Cette remarque permet d'exprimer simplement (d) en fonction des coordonnées de Plücker de (D). Pour cela, considérons la représentation (1.1) de (d) par le triplet unitaire $(a, b, c)^T$ et un point $\boldsymbol{p}$ de cette droite image. Ce point se trouvant sur (d), il se trouve donc dans le plan d'interprétation. Il est donc perpendiculaire au vecteur $\mathbf{h}$:

$$\boldsymbol{p}^T \mathbf{h} = 0$$

Cette expression est similaire à (1.2) et, par conséquent, $\mathbf{h}$ et $(a, b, c)^T$ sont proportionnels.

Comme le triplet $(a, b, c)^T$ est unitaire, on peut écrire cette proportionnalité comme $\mathbf{h} = \pm\|\mathbf{h}\|(a, b, c)^T$. Notons h la norme de $\mathbf{h}$ et $\underline{\mathbf{h}}$ son vecteur unitaire.

Prenons de plus la convention que (D) et (d) sont orientées de la même manière. Alors, on a :

$$\underline{\mathbf{h}} = (a, b, c)^T \tag{1.10}$$
$$\mathbf{h} = h\,\underline{\mathbf{h}} \tag{1.11}$$

Les coordonnées de la droite (d), dans le repère caméra, sont donc les composantes du vecteur $\underline{\mathbf{h}}$.

1.2.3 Utilisation en asservissement visuel

Asservissement 2D Les rares travaux [Cha90, Mot92, Deb96] portant sur l'asservissement visuel monoculaire à partir de droites, toutes basées sur la même approche initiale, mélangent la représentation (ρ, θ) des droites 2D et la représentation par intersection de plans des droites 3D. Nous verrons, au fur et à mesure de cette partie, combien cela est peu commode, notamment qu'elle induit une commande complètement couplée. En tout cas, on voit dès à présent le manque d'homogénéité entre ces deux représentations.

Asservissement stéréo De son côté, Hager [Hag97] représente, avec un système stéréoscopique, les droites 2D par le triplet des coefficients de leur équation normalisé sur les deux premières coordonnées (1.4), et les droites 3D par le couple point-direction (1.7). Ici aussi, apparaît une hétérogénéité qui va compliquer les choses par la suite. Pourtant, « le coup passa si près que le chapeau tomba » [Hug59].

En effet, travaillant avec un système stéréo fixe, il se place dans un repère arbitraire du monde dans lequel il exprime le couple point-direction $(\boldsymbol{P}, \underline{\mathbf{u}}_0)$ ainsi que la pose $(\mathbf{R}_i, \mathbf{t}_i)$ de chacune de ses caméras. Il exprime alors la droite projetée dans l'image i par le triplet normalisé $\mathbf{l}_i$ (1.4) obtenu via deux équations :

$$\boldsymbol{L'} = \mathbf{R}_i(\underline{\mathbf{u}}_0 \times (\boldsymbol{P} - \mathbf{t}_i)) \tag{1.12}$$
$$\mathbf{l}_i = \frac{\boldsymbol{L'}}{\sqrt{\boldsymbol{L'}_x^2 + \boldsymbol{L'}_y^2}} \tag{1.13}$$

Réécrivons la première de ces deux équations :

$$\boldsymbol{L'} = -\left(\mathbf{R}_i(\boldsymbol{P} \times \underline{\mathbf{u}}_0) + As(-\mathbf{R}_i\mathbf{t}_i)\mathbf{R}_i\underline{\mathbf{u}}_0\right)$$

et l'on voit apparaître les coordonnées de Plücker $(\underline{\mathbf{u}}_0, \mathbf{h}_0 = \boldsymbol{P} \times \underline{\mathbf{u}}_0)$, exprimées dans le repère du monde. Citons Navab [NFV93] pour se remémorer l'expression du changement des coordonnées de Plücker d'une droite par un déplacement euclidien $(\mathbf{R}, \mathbf{t})$ d'une position initiale $(\underline{\mathbf{u}}_0, \mathbf{h}_0)$ vers une position finale $(\underline{\mathbf{u}}, \mathbf{h})$:

$$\begin{pmatrix} \underline{\mathbf{u}} \\ \mathbf{h} \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{0}_{3\times3} \\ As(\mathbf{t})\mathbf{R} & \mathbf{R} \end{pmatrix} \begin{pmatrix} \underline{\mathbf{u}}_0 \\ \mathbf{h}_0 \end{pmatrix}$$

qui n'est autre que le changement de base d'un torseur [SLBE91, p. 49], puisque droites et torseurs sont intimement liés (Annexe A). Les déplacements euclidiens formant un groupe de Lie, la matrice de changement de coordonnées est aussi appelée *opérateur adjoint* (ou *tangent operator*, en anglais [Mac90]).

Comparons cette expression et celle de $\boldsymbol{L'}$ pour réaliser que $\boldsymbol{L'}$ n'est autre, au signe près, que la deuxième composante $\mathbf{h}$ des coordonnées de Plücker de la droite exprimées dans le repère caméra.

Par conséquent, si, au lieu de diviser $\boldsymbol{L'}$ par la norme de ses deux premiers coefficients pour obtenir $\mathbf{l}_i$, il avait divisé par $\|\boldsymbol{L'}\|$, il aurait obtenu $\mathbf{l}_i = \underline{\mathbf{h}}$. Nous verrons la conséquence de ce choix dans le calcul du mouvement.

1.2.4 Notre représentation

Nous suggérons une représentation des droites qui soit homogène aussi bien dans l'image que dans l'espace. Pour cela, nous nous basons sur les coordonnées de Plücker qui possèdent cette qualité.

Soit donc une droite (D) de l'espace de coordonnées de Plücker $(\mathbf{h}, \underline{\mathbf{u}})$ et (d), sa projection dans l'image, de coordonnées $\underline{\mathbf{h}}$.

Remarquons que le vecteur $\mathbf{h}$ s'interprète comme le produit du vecteur $\underline{\mathbf{h}}$, normal au plan d'interprétation, et du scalaire h. Ce scalaire peut être considéré comme une profondeur. En effet, $h = \|\mathbf{h}\| = \|\boldsymbol{H} \times \underline{\mathbf{u}}\|$ où $\boldsymbol{H}$ est le point de (D) le plus proche de l'origine. Le scalaire h est donc la distance orthogonale de l'origine à la droite (D). Il devient une profondeur dès lors que l'on considère l'ensemble des droites du plan d'interprétation, parallèles à $\underline{\mathbf{u}}$.

Afin de bien mettre en évidence, dans le vecteur $\mathbf{h}$, ces deux composantes – la profondeur (h) et la normale au plan d'interprétation $(\underline{\mathbf{h}})$ –, nous préférons les séparer et représenter la droite (D) par :

$$\mathsf{D} = \begin{pmatrix} \underline{\mathbf{h}} \\ \underline{\mathbf{u}} \\ h \end{pmatrix} \tag{1.14}$$

Ainsi, nous obtenons une interprétation « séquentielle » des coordonnées de Plücker (voir Figure 1.2) : la droite (D) est la droite 3D du plan d'interprétation de (d) — défini par son vecteur normal $\underline{\mathbf{h}}$ —, de direction $\underline{\mathbf{u}}$ et située à une profondeur h.

Cette représentation est redondante, ce qui est en général gênant puisqu'il faut alors tenir compte de contraintes, en l'occurence :

$$\begin{aligned} \|\underline{\mathbf{h}}\| &= 1 \\ \|\underline{\mathbf{u}}\| &= 1 \\ \underline{\mathbf{h}}^T \underline{\mathbf{u}} &= 0 \end{aligned}$$

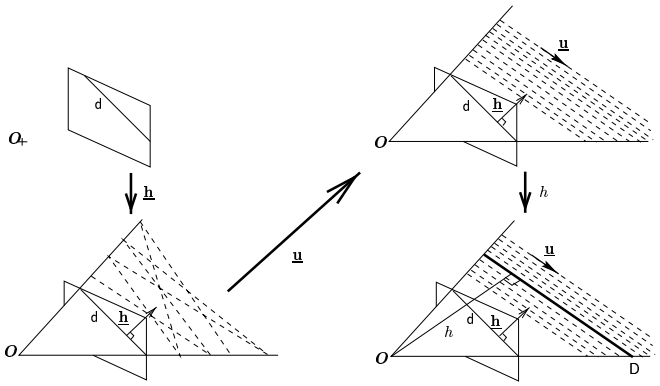


FIG. 1.2: *Interprétation géométrique « séquentielle » des coordonnées de Plücker d'une droite.*

Néanmoins, cette expression est plaisante pour plusieurs raisons. D'abord, elle isole bien les différentes contraintes qui caractérisent une droite. Ensuite, son interprétation correspond à la chaîne de traitement suivie en vision pour déterminer une droite 3D à partir d'une ou plusieurs images. En effet, cette chaîne se décompose en :

1. extraction d'une droite de l'image, identique à la détermination de $\underline{\mathbf{h}}$;
2. calcul d'orientation, équivalente à la détermination de $\underline{\mathbf{u}}$;
3. calcul de position, correspondant à la détermination de h.

Enfin, cette représentation permet de définir des *coordonnées de Plücker binormées* (CPN) qui sont le couple $(\underline{\mathbf{u}}, \underline{\mathbf{h}})$. On peut alors la considérer comme l'union de coordonnées de Plücker binormées et d'un facteur d'échelle positif. Une telle séparation s'avérera plus pratique dans la suite de ce travail puisque nous serons amenés à « oublier » la profondeur et qu'elle permettra la définition d'une commande partiellement découplée.

1.3 Mouvement apparent d'une droite

Dans ce paragraphe, nous nous intéressons au mouvement apparent , dans l'espace et dans l'image, d'une droite fixe de l'espace lorsque la caméra qui l'observe se déplace continûment. Nous noterons $\boldsymbol{\tau} = \begin{pmatrix} \mathbf{V} \\ \mathbf{\Omega} \end{pmatrix}$ le torseur cinématique, contenant la vitesse linéaire instantanée $\mathbf{V}$ et la vitesse angulaire instantanée $\mathbf{\Omega}$ du repère attaché à la caméra.

Le mouvement apparent d'une droite (D) fixe de l'espace s'écrit comme la dérivée temporelle des coordonnées de sa projection (d). Celle-ci s'exprime en fonction des

coordonnées actuelles de (D) et du torseur cinématique. Elle dépend donc de la représentation choisie pour les droites.

Nous verrons dans un premier temps pourquoi les représentations utilisées dans le passé en asservissement visuel ne fournissent pas des expressions aisément exploitables. À l'opposé, notre représentation fournit une expression simple dont nous tirerons parti dans le chapitre 2, consacré à la commande.

1.3.1 Limitations des approches précédentes

Asservissement 2D Le mouvement apparent d'une droite (D) de l'espace, dont la projection (d) est représentée par le couple (ρ, θ), s'exprime par le vecteur $(\dot{\rho}, \dot{\theta})$. Ce dernier est relié au mouvement de la caméra $\boldsymbol{\tau}$ par le biais d'une matrice d'interaction [Cha90] :

$$\begin{pmatrix} \dot{\theta} \\ \dot{\rho} \end{pmatrix} = \mathbf{L}_{2d}^T \begin{pmatrix} \mathbf{V} \\ \mathbf{\Omega} \end{pmatrix}$$

avec

$$\begin{aligned} \mathbf{L}_{2d}^T &= \begin{pmatrix} \lambda_\theta \cos\theta & \lambda_\theta \sin\theta & -\lambda_\theta \rho & -\rho\cos\theta & -\rho\sin\theta & -1 \\ \lambda_\rho \cos\theta & \lambda_\rho \sin\theta & -\lambda_\rho \rho & (1+\rho^2)\sin\theta & -(1+\rho^2)\cos\theta & 0 \end{pmatrix} \quad (1.15) \\ \lambda_\rho &= (a_i \rho \cos\theta + b_i \rho \sin\theta + c_i)/d_i \\ \lambda_\theta &= (a_i \sin\theta - b_i \cos\theta)/d_i \end{aligned}$$

Dans la matrice d'interaction $\mathbf{L}_{2d}^T$ apparaissent bien sûr les coordonnées courantes (ρ, θ) de la droite (d). À l'instar de la matrice d'interaction associée à un point, la matrice $\mathbf{L}_{2d}^T$ contient aussi des informations tridimensionnelles sur la position dans l'espace de (D). Ici, elles prennent la forme d'une dépendance de $\mathbf{L}_{2d}^T$ aux coefficients (a_i, b_i, c_i, d_i) de l'équation de celui des deux plans définissant (D) qui ne passe pas par l'origine.

Ces relations complexes n'offrent pas d'ouverture pour une analyse globale du mouvement apparent en fonction du mouvement de la caméra. De plus, elles ne se prêtent guère plus à une interprétation géométrique.

Asservissement stéréo Faute d'avoir remis en cause sa représentation des droites par le couple point-direction $(\boldsymbol{P}, \underline{\mathbf{u}})$, Hager [Hag97] s'enferre dans des expressions complexes pour exprimer le mouvement d'une droite image $\mathbf{l} = (a, b, c)^T$. Ainsi, au lieu de simplement dériver des coordonnées de Plücker, il aboutit à la matrice d'interaction suivante :

$$\mathbf{J} = \mathbf{N}(\mathbf{l})\mathbf{J}'(\boldsymbol{P}, \underline{\mathbf{u}})$$

où la matrice $\mathbf{N}(\mathbf{l})$ est le Jacobien de son opération de normalisation (1.13) :

$$\mathbf{N}(\mathbf{l}) = \frac{1}{(a^2+b^2)^{3/2}} \begin{pmatrix} a^2 & -ab & 0 \\ -ab & b^2 & 0 \\ -ca & -cb & (a^2+b^2) \end{pmatrix}$$

et $\mathbf{J}'(\boldsymbol{P}, \underline{\mathbf{u}})$ est le Jacobien associé à (1.12) :

$$\mathbf{J}'(\boldsymbol{P}, \underline{\mathbf{u}}) = \mathbf{R}_i \begin{pmatrix} As(\underline{\mathbf{u}}) & As(\underline{\mathbf{u}})As(-\boldsymbol{P}) + As(\boldsymbol{P} - \mathbf{t}_i)As(\underline{\mathbf{u}}) \end{pmatrix}$$

C'est bien compliqué et n'aide pas beaucoup à la compréhension. Si, en manipulant l'expression de $\mathbf{J}'(\boldsymbol{P}, \underline{\mathbf{u}})$, on peut aboutir au Jacobien des coordonnées de Plücker (modulo un changement de base du torseur cinématique), la matrice $\mathbf{N}(\mathbf{l})$ n'est en revanche pas aisément utilisable. Nous allons immédiatement voir que notre représentation des droites, qui est basée sur l'autre choix de normalisation des coordonnées d'une droite image, offre des propriétés plus sympathiques.

1.3.2 Avec notre représentation

Considérons une droite (D) fixe de coordonnées $(\underline{\mathbf{h}}^T, \underline{\mathbf{u}}^T, h)^T$ (exprimées dans le repère caméra) se projetant en la droite (d) de coordonnées $\underline{\mathbf{h}}$ (exprimées aussi dans le repère caméra). Le mouvement de (D) relatif au repère mobile de la caméra est donc le vecteur des dérivées de chacune des composantes de ses coordonnées $(\dot{\underline{\mathbf{h}}}^T, \dot{\underline{\mathbf{u}}}^T, \dot{h})^T$.

Retrouvons les deux premières composantes à partir des dérivées $(\dot{\mathbf{u}}, \dot{\mathbf{h}})$ des coordonnées de Plücker $(\mathbf{u}, \mathbf{h})$ d'une droite 3D [RE87] :

$$\dot{\underline{\mathbf{u}}} = \boldsymbol{\Omega} \times \underline{\mathbf{u}} \tag{1.16}$$

$$\dot{\mathbf{h}} = \boldsymbol{\Omega} \times \mathbf{h} + \mathbf{V} \times \underline{\mathbf{u}} \tag{1.17}$$

Navab [NFV93] en déduit l'équation de ce qu'il appelle le *line motion field*. Ce n'est autre que la relation entre le mouvement de la caméra et le mouvement apparent de la droite (D) dans l'image. Ce dernier est le mouvement de (d) dans l'image et est donc représenté par la dérivée $\dot{\underline{\mathbf{h}}}$ des coordonnées de (d), que Navab écrit de deux façons :

$$\dot{\underline{\mathbf{h}}} = \boldsymbol{\Omega} \times \underline{\mathbf{h}} + \frac{1}{h}(\mathbf{I}_3 - \underline{\mathbf{h}}\underline{\mathbf{h}}^T)(\mathbf{V} \times \underline{\mathbf{u}}) \tag{1.18}$$

$$\dot{\underline{\mathbf{h}}} = \boldsymbol{\Omega} \times \underline{\mathbf{h}} - \frac{\mathbf{V}^T\underline{\mathbf{h}}}{h}(\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \tag{1.19}$$

Le passage de l'une à l'autre se fait en remarquant que la matrice $(\mathbf{I}_3 - \underline{\mathbf{h}}\,\underline{\mathbf{h}}^T)$ est un opérateur de projection orthogonale sur le plan perpendiculaire à $\underline{\mathbf{h}}$ car :

$$\begin{aligned} (\mathbf{I}_3 - \underline{\mathbf{h}}\,\underline{\mathbf{h}}^T)\underline{\mathbf{h}} &= 0 \\ (\mathbf{I}_3 - \underline{\mathbf{h}}\,\underline{\mathbf{h}}^T)\mathbf{u} &= \mathbf{u} \\ (\mathbf{I}_3 - \underline{\mathbf{h}}\,\underline{\mathbf{h}}^T)\mathbf{u} \times \underline{\mathbf{h}} &= \mathbf{u} \times \underline{\mathbf{h}} \end{aligned} \tag{1.20}$$

Suite à ces travaux, il reste à calculer la variation de la profondeur $\dot{h}$ pour compléter l'expression du mouvement de (D). De la définition de $h = \sqrt{\mathbf{h}^T\mathbf{h}}$, on tire

$$\dot{h} = \frac{\mathbf{h}^T\dot{\mathbf{h}}}{\sqrt{\mathbf{h}^T\mathbf{h}}}$$

que l'on simplifie en $\dot{h} = \underline{\mathbf{h}}^T\dot{\mathbf{h}}$. Remplaçant alors, $\dot{\mathbf{h}}$ par son expression (1.17), on obtient alors :

$$\dot{h} = \underline{\mathbf{h}}^T(\boldsymbol{\Omega} \times \mathbf{h} + \mathbf{V} \times \underline{\mathbf{u}})$$

En développant ce produit scalaire et en utilisant la propriété d'invariance par permutation de ses arguments du produit mixte, on obtient finalement :

$$\dot{h} = \mathbf{V}^T(\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \tag{1.21}$$

Pour la beauté du geste[1], nous pouvons finalement exprimer le mouvement instantané relatif d'une droite 3D fixe par rapport à une caméra soumise au torseur cinématique $\boldsymbol{\tau} = \left(\begin{smallmatrix}\mathbf{V}\\ \boldsymbol{\Omega}\end{smallmatrix}\right)$ par le biais d'une matrice d'interaction en réécrivant (1.19), (1.16) et (1.21) sous forme matricielle :

$$\begin{pmatrix}\dot{\underline{\mathbf{h}}}\\ \dot{\underline{\mathbf{u}}}\\ \dot{h}\end{pmatrix} = \begin{pmatrix}-\frac{1}{h}(\underline{\mathbf{u}} \times \underline{\mathbf{h}})\underline{\mathbf{h}}^T & -As(\underline{\mathbf{h}})\\ \mathbf{0}_{3\times3} & -As(\underline{\mathbf{u}})\\ (\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T & \mathbf{0}_{1\times3}\end{pmatrix}\begin{pmatrix}\mathbf{V}\\ \boldsymbol{\Omega}\end{pmatrix} \tag{1.22}$$

En résumé, nous avons obtenu deux expressions cohérentes : celle du mouvement relatif de (D) lorsque la caméra se déplace et celle du mouvement apparent de sa projection (d) dans l'image.

1.4 Alignement de droites

Rappelons que l'asservissement visuel monoculaire est une technique pour positionner dans l'espace euclidien une caméra par rapport à une scène. Pour ce faire, on utilise des primitives géométriques de l'espace, observées par la caméra, que l'on cherche à amener dans une configuration cible (explicite ou non) dans l'espace.

Hormis le cas de l'asservissement 3D, où l'on travaille directement sur la pose 3D de la caméra par rapport à la scène, on est amené à superposer chaque primitive sur une position cible.

Lorsque l'on utilise des points, on a en théorie deux alignements possibles : superposition sur des points cibles de l'espace ou superposition de leurs projections sur des points cibles dans l'image. Toutes deux fournissent la même projection dans l'image.

1. Car nous n'utiliserons plus cette notation dans la suite de ce document

Néanmoins, en pratique, on ne réalise pas la première car elle nécessite un calcul de profondeur, souvent synonyme de longs calculs de pose ou de reconstruction.

On se contente en fait de l'alignement dans l'image : on amène les projections des points 3D en des positions cibles définies dans l'image. La contrainte de rigidité, un nombre suffisant de points et une bonne loi de commande font alors le reste, à savoir la superposition des points 3D sur leurs positions cibles induites dans l'espace exprimées dans le repère de la caméra.

En ne considérant qu'un point $\boldsymbol{P}$ que signifie cet alignement dans l'image? Habituellement, on se cantonne à considérer que $\boldsymbol{P}$ se trouve sur la bonne ligne de vue, ce qui reste une interprétation bidimensionnelle. Mais si l'on se place en 3D, on réalise que $\boldsymbol{P}$ est amené en un point $\boldsymbol{P'}$, équivalent à la profondeur près au point $\boldsymbol{P}*$ où l'on souhaiterait qu'il se trouve physiquement (voir Figure 1.3).

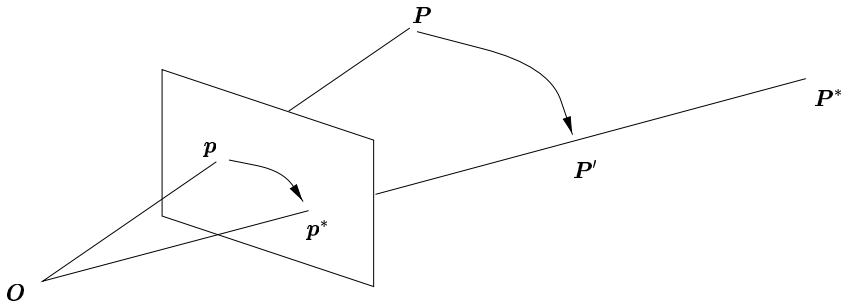


FIG. 1.3: *Interprétation* 3D *de l'alignement* 2D *d'un point.*

Ces deux interprétations sont équivalentes, mais le sont-elles aussi pour les droites?

À nouveau, les choses se corsent lorsque l'on passe aux droites. En effet, on peut distinguer désormais 4 types d'alignement qui fournissent la même droite projetée. Pour cela reprenons notre représentation d'une droite 3D $(\underline{\mathbf{h}}^T, \underline{\mathbf{u}}^T, h)^T$. La contrainte imposant une projection image désirée consiste à fixer $\underline{\mathbf{h}}$. Il reste alors à choisir $\underline{\mathbf{u}}$ et h. Selon que l'on se donne une valeur pour l'un et/ou pour l'autre ou non permet d'obtenir les 4 types d'alignement (voir Figure 1.4).

Lorsque l'on n'impose aucune contrainte supplémentaire (c.-à-d. que la direction $\underline{\mathbf{u}}$ et la profondeur h sont libres), on obtient l'alignement 2D classique, utilisé par Chaumette. Lorsque l'on contraint simultanément $\underline{\mathbf{u}}$ et h, on obtient un alignement 3D. Lorsque seule la profondeur h est fixée, on impose à la droite 3D d'être tangente au cercle situé dans le plan d'interprétation défini par $\underline{\mathbf{h}}$, centré en l'origine et de rayon h. Enfin, lorsque l'on libère la profondeur mais pas la direction, on impose à la droite 3D de se trouver dans le faisceau des droites parallèles à $\underline{\mathbf{u}}$ du plan d'interprétation.

L'alignement à profondeur fixée ne semble pas avoir de véritable intérêt pratique. L'alignement 3D nécessite un calcul de pose ou une reconstruction et n'est

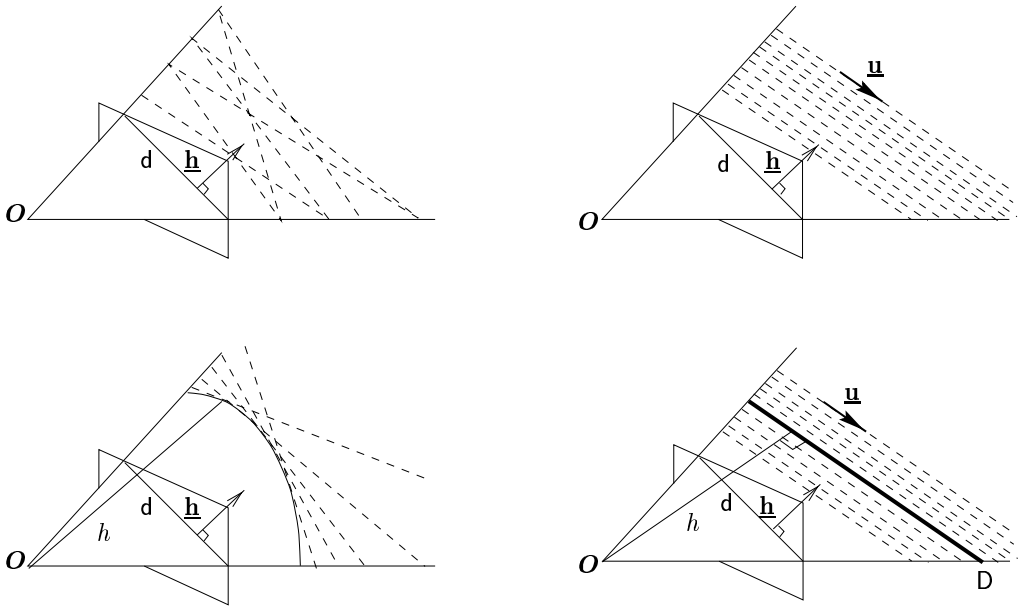


FIG. 1.4: *Interprétation* 3D *de l'alignement* 2D *d'une droite.*

pas forcément très intéressant. En revanche, l'alignement classique a d'ores et déjà fait ses preuves. Reste l'alignement à direction fixée, ou *alignement en coordonnées de Plücker binormées.*

On notera d'abord que le faisceau de droites qu'il définit est la variété linéaire d'ordre 1 des droites équivalentes à une profondeur près. On retrouve ainsi le pendant de l'interprétation alternative de l'alignement dans l'image de deux points.

De manière plus terre-à-terre, on remarquera aussi qu'il n'est pas forcément nécessaire de passer par un calcul de pose pour obtenir la direction d'une droite à partir d'une seule image. En effet, de nombreuses applications permettent son obtention directe.

Le premier exemple est le suivi de route. Dans cette application, on fait l'hypothèse que la route est droite et on suit soit la ligne centrale, soit le bord de la route. Géométriquement, on observe dans l'image la projection d'une droite 3D « tracée » sur le sol. Celui-ci est un plan dont on connaît parfaitement la normale $\underline{\mathbf{n}}$ puisqu'il est parallèle au châssis. La droite est donc perpendiculaire à cette normale ainsi qu'à la normale au plan d'interprétation défini par la droite image, qui n'est autre que les coordonnées $\underline{\mathbf{h}}$ de cette dernière. Ces deux plans n'étant pas parallèles, leur intersection est une droite orientée par $\underline{\mathbf{u}} = \underline{\mathbf{n}} \times \underline{\mathbf{h}}$. On a donc de façon très simple les deux premières composantes ($\underline{\mathbf{h}}$ et $\underline{\mathbf{u}}$) de la droite 3D. En revanche, la profondeur est moins digne de confiance puisqu'elle dépend de la hauteur du châssis par rapport au sol, qui n'est pas stable (irrégularités du sol, pression des pneus, etc.).

Un autre exemple est l'atterrissage d'un avion. De la même manière que précé-

demment, on a l'orientation du sol grâce à l'horizon artificiel, très stable grâce aux propriétés physiques des gyroscopes. L'altitude est, elle, moins stable puisqu'elle est soumise aux trous d'air, mauvais étalonnage des altimètres, etc.

Un autre cas simple de calcul de la direction d'une droite 3D à partir d'une seule image intervient lorsque l'on observe deux (ou plus) droites 3D parallèles. Dans ce cas, leur direction commune n'est autre que celle de la droite de vue passant par leur point de fuite (voir Figure 1.5). Un exemple, autre que les deux précédents, d'un tel cas est le passage d'une porte par un robot mobile car on dispose alors des deux montants parallèles verticaux.

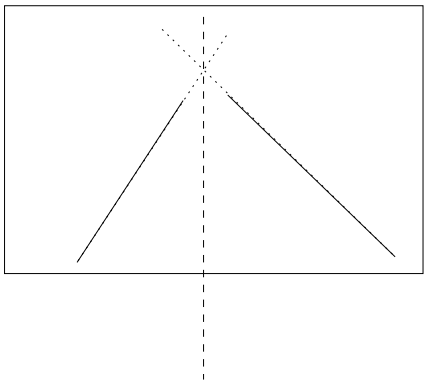


FIG. 1.5: *La direction de deux droites parallèles (trait plein) se déduit directement de l'image.*

En résumé, nous disposons de deux types d'alignement de droite utilisables en pratique : l'alignement classique qui laisse deux degrés de liberté à la droite

$$\underline{\mathbf{h}} = \underline{\mathbf{h}}^* \tag{1.23}$$

et l'alignement en coordonnées de Plücker binormées qui n'en laisse qu'un

$$\begin{aligned} \underline{\mathbf{h}} &= \underline{\mathbf{h}}^* \\ \underline{\mathbf{u}} &= \underline{\mathbf{u}}^* \end{aligned} \tag{1.24}$$

C'est alors une affaire de goût de choisir l'une ou l'autre. Nous allons voir toutefois dans le chapitre suivant que l'utilisation du nouvel alignement permet de simplifier la commande puisqu'il permet un découplage partiel entre rotation et translation.

1.5 Conclusion

En conclusion de ce chapitre, nous avons visité les différentes représentations de droite et en avons tiré une nouvelle, dérivée des coordonnées euclidiennes de

Plücker. Celle-ci nous a permis de reformuler la relation entre le mouvement de la caméra et la vitesse apparente d'une droite dans l'image. Elle nous a aussi permis de mettre à jour plusieurs types d'alignement de droites, ce qui représente un pas en avant vers ce qu'Hager appelle « *a "line-to-line" positioning primitive* » [Hag97]. Ces deux reformulations nous seront bien utiles dans le chapitre suivant pour définir une commande analytique, géométriquement interprétable et subjectivement esthétique.

Chapitre 2

Commande

2.1 Introduction

Dans ce chapitre, nous allons désormais pouvoir nous consacrer à la génération d'une loi de commande. Il s'agit donc de définir un algorithme qui régule une erreur entre une donnée de référence et une donnée courante à zéro. Cette erreur s'exprime, en général, comme la distance entre ces deux données.

Lorsque l'on fait de l'asservissement visuel en extrayant des points des images, on cherche à superposer un ensemble de points de l'image sur un autre, correspondant à la superposition d'un nuage de points de l'espace sur un autre. La distance dans l'image est alors « homogène » à celle dans l'espace puisqu'elles sont naturellement définies comme les distances euclidiennes dans leur espace respectif.

Dans le cas des droites, en revanche, le choix de la distance est moins naturel. En effet, la variété, de dimension 4, des droites dans $\mathbb{R}^3$ ne possède pas de distance naturelle, non plus que celle, de dimension 2, des droites dans $\mathbb{R}^2$. Cela est mis en évidence au chapitre précédent par l'existence de représentations multiples de ces droites. Ce problème intrinsèque est d'ailleurs analogue à celui que l'on rencontre dans $SE(3)$, lorsque l'on cherche une distance entre deux déplacements euclidiens.

Il n'existe donc pas de choix naturel de commande pour les droites. Cependant, les coordonnées de Plücker nous offrent une opportunité de cohérence entre droites de l'image et droites de l'espace, qui nous permettra d'obtenir une commande explicite. Étant donné qu'elle n'est pas intuitive, en particulier parce qu'interviennent des non-linéarités, nous allons procéder de manière constructive.

Ainsi, nous passerons d'abord en revue les commandes que l'on est tenté de choisir par mimétisme avec celles basées sur l'utilisation de points. Elles dépendent du choix qui est fait pour la représentation des droites et du choix de l'erreur que

l'on cherche à réguler à zéro. Nous verrons quels en sont les inconvénients, majeurs ou subjectifs.

Nous aborderons ensuite la loi de commande finale, basée sur l'alignement en coordonnées de Plücker binormées. Elle se place dans un espace mixte entre le 2D et le 3D à l'instar de la commande en 2D1/2 de Malis[Mal98]. Afin d'éviter une course aux fractions de D, nous préférons conserver le nom complet de *commande en coordonnées de Plücker binormées*, qu'il nous arrivera d'abréger en *commande CPN*.

La commande en coordonnées de Plücker binormées possède deux caractéristiques majeures. Tout d'abord, tirant parti d'un découplage partiel entre rotation et translation qui apparaît dans les équations du mouvement (1.16) et (1.18), elle consiste en une inversion analytique exacte des équations du mouvement. Elle ne nécessite donc aucune inversion numérique de matrice.

La deuxième caractéristique est que la commande en coordonnées de Plücker binormées est conçue pour ne pas avoir à manipuler la profondeur, qui est une grandeur peu fiable en vision.

Rappelons avant d'aller plus loin que nous considérons que la caméra est étalonnée. Par conséquent, on ne verra pas apparaître ses paramètres intrinsèques ni la transformation pince-caméra dans la définition des lois de commandes.

2.2 Commandes intuitives

Soient (D_i), $i = 1..n$, les droites 3D observées par la caméra et (d_i), leurs projections respectives dans l'image.

2.2.1 Rappel de la commande 2D

La première commande que l'on est amené à envisager est la commande 2D de Chaumette [Cha90]. Elle consiste à prendre pour vecteur d'état $\mathbf{s}$, le vecteur composé des coordonnées réduites (ρ_i, θ_i) des droites images (d_i). Alors, on peut exprimer le mouvement apparent de ces droites par :

$$\dot{\mathbf{s}} = \frac{d}{dt}\begin{pmatrix}\rho_1\\ \theta_1\\ \vdots\\ \rho_n\\ \theta_n\end{pmatrix} = \mathbf{L}^T\begin{pmatrix}\mathbf{V}\\ \mathbf{\Omega}\end{pmatrix} \quad \text{avec} \quad \mathbf{L}^T = \begin{pmatrix}— \ \mathbf{L}_1^T \ —\\ \vdots\\ — \ \mathbf{L}_n^T \ —\end{pmatrix} \tag{2.1}$$

où les matrices d'interactions $\mathbf{L}_i^T$ sont définies par (1.15).

L'espace d'état étant linéaire, on peut alors définir une l'erreur simple suivante :

$$\mathbf{e} = \mathbf{C}(\mathbf{s} - \mathbf{s}^*) = \mathbf{C} \begin{pmatrix} \rho_1 - \rho_1^* \\ \theta_1 - \theta_1^* \\ \vdots \\ \rho_n - \rho_n^* \\ \theta_n - \theta_n^* \end{pmatrix}$$

où $\mathbf{C}$ est une matrice de combinaison $(6 \times 2n)$ à choisir convenablement. En choisissant le comportement exponentiel décroissant :

$$\dot{\mathbf{e}} = -\lambda \mathbf{e},$$

on aboutit à une loi de commande de la forme :

$$\begin{pmatrix} \mathbf{V} \\ \mathbf{\Omega} \end{pmatrix} = -\lambda {\mathbf{L}^T}^+ (\mathbf{s} - \mathbf{s}^*)$$

Cependant, comme les matrices d'interaction $\mathbf{L}_i^T$ sont potentiellement « pleines » (c.-à-d. sans blocs identiquement nuls), la matrice ${\mathbf{L}^T}^+$ fait apparaître des couplages entre les six composantes du torseur de commande $(\mathbf{V}, \mathbf{\Omega})$. Ces couplages peuvent alors générer un mouvement du robot, dans l'espace cartésien, selon des trajectoires peu rectilignes.

Similairement, on peut définir une commande 2D avec notre représentation en n'utilisant que l'équation (1.18). Il faut alors réguler à zéro l'erreur $e = \frac{1}{2}\|\underline{\mathbf{h}} \times \underline{\mathbf{h}}^*\|^2$. Ceci dit, cette commande possèdera le même genre de trajectoires peu rectilignes. On peut alors envisager de faire une commande 3D.

2.2.2 Commande 3D

À partir des droites (d_i) extraites des images et des paramètres intrinsèques de la caméra qui les observe, on peut procéder à un calcul de pose [DRLR89, Che91a]. On peut alors asservir le robot sur la différence entre la pose courante et la pose désirée. On obtient donc une commande dans $SE(3)$, à choisir parmi celles disponibles [MDGD97].

L'avantage d'un tel choix de commande est qu'il suffit de remplacer le calcul de pose à partir de points par un calcul de pose à partir de droites pour disposer du même comportement du robot dans l'espace. Les inconvénients en sont qu'il n'existe pas de résultats concernant la robustesse du calcul de pose par rapport aux erreurs de mesure, du fait que les algorithmes de calcul de pose impliquent la résolution de polynômes de degré 8 sans solution explicite, et qu'il est donc difficile d'obtenir des résultats de robustesse de la commande. Enfin, et surtout, on conserve l'inconvénient

d'une commande dans $SE(3)$ qui est de ne rien imposer aux trajectoires dans l'image et donc de risquer de perdre de vue les droites.

À l'opposé, on peut être tenté d'utiliser un vecteur d'état dans l'espace 3D. Dans le cas des droites, il pourrait alors s'agir d'un vecteur contenant les coordonnées de Plücker des droites 3D observées. Toutefois, cela se ramènerait le plus souvent au cas précédent[1] puisque les coordonnées de Plücker seraient obtenues par calcul de pose, à moins de disposer d'autres moyens de mesure qu'une unique caméra.

On est donc amené à chercher une commande mélangeant 2D et 3D.

2.3 Commande CPN

Nous verrons deux variantes de la commande en coordonnées de Plücker binormées. Elles diffèrent par le choix de l'erreur à réguler à 0, celle de la deuxième variante étant la projection sur un espace adéquat de l'erreur utilisée dans la première variante. Cela a des implications sur les théorèmes de convergence, que lecteur pressé ira consulter : Théorèmes 1 (p. 65), 2 (p. 66) et 3 (p. 69).

2.3.1 Rappel

La commande en coordonnées de Plücker binormées est basée sur la représentation (1.14) des droites que nous avons choisie au chapitre précédent et que nous rappelons ici :

$$\mathsf{D}_i = \begin{pmatrix} \underline{\mathbf{h}}_i \\ \underline{\mathbf{u}}_i \\ h_i \end{pmatrix}$$

où $\underline{\mathbf{h}}_i$ est aussi la représentation de la projection (d_i) de (D_i) dans l'image, $\underline{\mathbf{u}}_i$ est la direction 3D de la droite et h_i est sa profondeur.

Nous avons pour but de réaliser l'alignement en coordonnées de Plücker binormées (1.24), que nous rappelons aussi :

$$\begin{matrix} \underline{\mathbf{h}}_i = \underline{\mathbf{h}}_i^* \\ \underline{\mathbf{u}}_i = \underline{\mathbf{u}}_i^* \end{matrix} \quad i = 1..n$$

où la notation avec astérisque représente la valeur désirée (ou consigne) et la notation sans astérisque représente la valeur courante. Remémorons-nous aussi les équations du mouvement (1.16), (1.18) et (1.21) associées à cette représentation pour chaque

1. Néanmoins, les trajectoires dans l'image et dans l'espace pourront être différentes de celles obtenues avec la commande dans $SE(3)$.

droite i :

$$\dot{\underline{\mathbf{u}}}_i = \boldsymbol{\Omega} \times \underline{\mathbf{u}}_i \tag{2.2}$$

$$\dot{\underline{\mathbf{h}}}_i = \boldsymbol{\Omega} \times \underline{\mathbf{h}}_i + \frac{1}{h_i}(\mathbf{I}_3 - \underline{\mathbf{h}}_i \underline{\mathbf{h}}_i^T)(\mathbf{V} \times \underline{\mathbf{u}}_i) \tag{2.3}$$

$$\dot{h}_i = \mathbf{V}^T(\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i) \tag{2.4}$$

2.3.2 Hypothèses de travail

Par la suite, nous nous placerons toujours sous l'hypothèse que les consignes sont réalisables. Cela signifie que la configuration des droites considérées reste fixe durant l'asservissement, qu'elle est identique à la configuration désirée et qu'il existe donc une position du robot qui vérifie la consigne. Nous supposerons aussi que l'espace de travail du robot est suffisamment régulier et dégagé pour atteindre une position vérifiant la consigne.

Enfin, nous ferons l'hypothèse supplémentaire que les mesures sont exactes. L'asservissement visuel étant connu pour être robuste aux erreurs de mesure, cela atténue le caractère contraignant de cette hypothèse.

2.3.3 Commande non projetée

Une droite

Plaçons-nous dans un premier temps dans le cas d'une seule droite $\mathsf{D} = (\underline{\mathbf{h}}^T, \underline{\mathbf{u}}^T, h)$ et abandonnons jusqu'à nouvel ordre les indices. Supposons de plus qu'on peut extraire cette droite de l'image et qu'il existe un moyen d'obtenir sa direction. En revanche, nous ne ferons aucune hypothèse quant à sa profondeur. Par conséquent, la commande en coordonnées de Plücker binormées ne doit dépendre que des coordonnées $\underline{\mathbf{h}}$ de la droite image, de la direction $\underline{\mathbf{u}}$ et de leurs valeurs désirées ($\underline{\mathbf{h}}^*$ et $\underline{\mathbf{u}}^*$).

Nous allons utiliser, comme nous l'avons déjà fait, une approche constructiviste : nous allons progressivement aboutir à la solution en raffinant progressivement une idée de départ assez simple.

Comme nous n'utiliserons pas la profondeur, les deux seules équations du mouvement qui nous intéressent sont :

$$\dot{\underline{\mathbf{u}}} = \boldsymbol{\Omega} \times \underline{\mathbf{u}} \tag{2.5}$$

$$\dot{\underline{\mathbf{h}}} = \boldsymbol{\Omega} \times \underline{\mathbf{h}} + \frac{1}{h}(\mathbf{I}_3 - \underline{\mathbf{h}}\underline{\mathbf{h}}^T)(\mathbf{V} \times \underline{\mathbf{u}}) \tag{2.6}$$

$$= \boldsymbol{\Omega} \times \underline{\mathbf{h}} - \frac{1}{h}\mathbf{V}^T\underline{\mathbf{h}}(\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \tag{2.7}$$

On remarque que la première permet de définir $\boldsymbol{\Omega}$ uniquement à partir de la direction.

Asservissement visuel à partir de droites

Dans la deuxième équation, tout semble intimement lié. De plus, il reste la profondeur h. Toutefois, si le premier terme s'annule, alors cette dernière n'intervient plus que sous forme d'un gain : $\dot{\underline{\mathbf{h}}} = \frac{1}{h}(\mathbf{I}_3 - \underline{\mathbf{h}}\underline{\mathbf{h}}^T)(\mathbf{V} \times \underline{\mathbf{u}}) = -\frac{1}{h}\mathbf{V}^T\underline{\mathbf{h}}(\underline{\mathbf{u}} \times \underline{\mathbf{h}})$.

Ce découplage entre commande en rotation et commande en translation est intéressant car il permet de tenir compte du fait qu'il n'y a pas de métrique commune pour le vecteur des vitesses linéaires et celui des vitesses angulaires. Ceci s'explique, de manière intuitive, par le fait que l'un s'exprime en m/s et l'autre en rad/s et, de manière plus formelle, par le fait que le groupe des déplacements rigides ($SE(3)$) ne possède pas de métrique [MLS94]. Ainsi, il est plus raisonnable de les définir indépendamment l'un de l'autre que simultanément via la pseudo-inverse d'un Jacobien. Enfin, et surtout, le découplage est intrinsèquement intéressant puisqu'il évite que la régulation à zéro de certaines erreurs (ici, l'erreur sur $\underline{\mathbf{h}}$) n'engendre des perturbations dans la régulation d'autres erreurs (ici, l'erreur sur $\underline{\mathbf{u}}$).

Nous allons donc étudier dans un premier temps, la commande en rotation seule, puis la commande en translation seule, et enfin, nous les regrouperons en une commande complète.

Commande en rotation — La commande en rotation vient trivialement, et en s'inspirant de [LB87], en choisissant l'erreur à réguler à 0 :

$$\mathbf{e}_u = \underline{\mathbf{u}} \times \underline{\mathbf{u}}^* \tag{2.8}$$

qui est la mesure classique de la distance entre deux vecteurs unitaires.

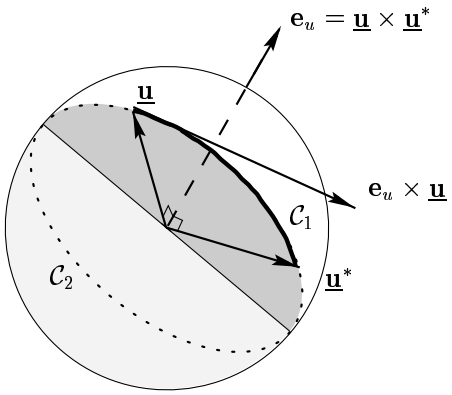


FIG. 2.1: *Géodésique et erreur entre deux vecteurs unitaires (voir texte).*

Interprétons géométriquement cette erreur (voir Figure 2.1). Le vecteur $\underline{\mathbf{u}}$ est un vecteur unitaire et est donc condamné à se déplacer sur la sphère unité. Par conséquent, l'erreur précédente est alors le vecteur perpendiculaire au plan vectoriel défini par $\underline{\mathbf{u}}$ et $\underline{\mathbf{u}}^*$ et le produit vectoriel $\mathbf{e}_u \times \underline{\mathbf{u}}$ s'interprète comme le vecteur tangent en $\underline{\mathbf{u}}$ au grand-cercle reliant $\underline{\mathbf{u}}$ à $\underline{\mathbf{u}}^*$. Les deux arcs de ce grand-cercle $\mathcal{C}_1$ et $\mathcal{C}_2$ entre

ces deux points sont appelées géodésiques et l'erreur est orientée de telle manière que le vecteur tangent soit dirigé vers la géodésique la plus courte $\mathcal{C}_1$. Enfin, la norme de l'erreur est la longueur de cette dernière, qui est aussi la distance la plus courte sur la sphère entre $\underline{\mathbf{u}}$ et $\underline{\mathbf{u}}^*$ [SLBE91].

Alors, nous avons le lemme suivant qui étend les résultats de Le Borgne [LB87] :

Lemme 1 (Lemme de convergence)

La commande en rotation définie par :

$$\begin{aligned} \mathbf{e}_u &= \underline{\mathbf{u}} \times \underline{\mathbf{u}}^* \\ \mathbf{\Omega} &= \mu_u \, \mathbf{e}_u, \mu_u > 0 \end{aligned}$$

est asymptotiquement stable si la condition initiale $\underline{\mathbf{u}}(t=0) \neq -\underline{\mathbf{u}}^*$ *est vérifiée. Cette condition représente un équilibre instable.*

Preuve :

On considère la fonction de Lyapunov :

$$L_u = \frac{1}{2}\|\underline{\mathbf{u}} - \underline{\mathbf{u}}^*\|^2$$

dont la dérivée est $\dot{L}_u = \dot{\underline{\mathbf{u}}}^T(\underline{\mathbf{u}} - \underline{\mathbf{u}}^*)$ et se simplifie en :

$$\dot{L}_u = -\dot{\underline{\mathbf{u}}}^T \underline{\mathbf{u}}^* \tag{2.9}$$

du fait de l'orthogonalité entre un vecteur unitaire et sa dérivée.

Lorsque l'on y insère la loi de commande de l'énoncé, l'expression précédente devient :

$$\dot{L}_u = -\mu_u((\underline{\mathbf{u}} \times \underline{\mathbf{u}}^*) \times \underline{\mathbf{u}})^T \underline{\mathbf{u}}^*$$

En développant le double produit vectoriel selon la règle

$$(\mathbf{a} \times \mathbf{b}) \times \mathbf{c} = \mathbf{a}^T\mathbf{c}\mathbf{b} - \mathbf{b}^T\mathbf{c}\mathbf{a},$$

on obtient finalement :

$$\dot{L}_u = -\mu_u(1 - (\underline{\mathbf{u}}^T \underline{\mathbf{u}}^*)^2)$$

Comme $\underline{\mathbf{u}}$ et $\underline{\mathbf{u}}^*$ sont unitaires, cette dérivée est donc négative si μ_u est positif. Elle ne s'annule que lorsque $\underline{\mathbf{u}} = \pm\underline{\mathbf{u}}^*$, c.-à-d. à l'équilibre et en la singularité $\underline{\mathbf{u}} = -\underline{\mathbf{u}}^*$, qui correspond au cas où L_u est maximale. Comme

$\dot{L}_u$ est négative, on n'atteindra donc pas cette singularité, à moins de s'y trouver à l'instant initial.

Par conséquent, en dehors de la situation initiale $\underline{\mathbf{u}}(t=0) = -\underline{\mathbf{u}}^*$, la fonction de Lyapunov L_u est strictement décroissante et s'annule au point de convergence. Donc, la commande est asymptotiquement stable.

On conçoit aisément que la condition initiale $\underline{\mathbf{u}}(t=0) = -\underline{\mathbf{u}}^*$ représente un équilibre instable, puisqu'une faible perturbation de $\underline{\mathbf{u}}(t=0)$ nous ramène en une position où $\dot{L}_u$ est strictement négative. □

Ainsi, l'erreur choisie par Le Borgne ne décroît que lorsque $\underline{\mathbf{u}}(t=0)$ et $\underline{\mathbf{u}}^*$ forment un angle aigu. Il obtient alors une convergence exponentielle de cette erreur. En revanche, l'erreur $\|\underline{\mathbf{u}}(t=0) - \underline{\mathbf{u}}^*\|$ décroît dès lors qu'ils ne sont pas opposés. Comme la commande n'est pas proportionnelle à cette erreur, elle ne sera décroissante que pour un écart angulaire inférieur à $\pi/2$ et accélérera jusqu'à ce que cet écart atteigne $\pi/2$ autrement.

Un autre résultat intéressant concerne le cas où $\underline{\mathbf{u}} \neq \underline{\mathbf{u}}^*$ et que les vecteurs $\underline{\mathbf{u}}$, $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ sont coplanaires. Alors on obtient la propriété suivante lors de l'activation de la commande en rotation précédente :

Lemme 2

Soient les conditions initiales suivantes (Figure 2.2) :

- *les vecteurs* $\underline{\mathbf{u}}(t=0)$, $\underline{\mathbf{h}}(t=0)$ *et* $\underline{\mathbf{h}}^*$ *sont coplanaires ;*
- *les vecteurs* $\underline{\mathbf{h}}(t=0)$ *et* $\underline{\mathbf{h}}^*$ *forment un angle aigu ;*
- *les vecteurs* $\underline{\mathbf{h}}(t=0)$ *et* $\underline{\mathbf{u}}^*$ *ne sont pas orthogonaux.*

Si ces conditions sont vérifiées, et sous l'hypothèse que la commande en rotation définie au Lemme 1 est parfaite (c.-à-d. n'engendre pas de translations parasites), alors cette dernière ne rompt pas la condition de coplanarité et converge asymptotiquement vers l'alignement en coordonnées de Plücker binormées ($\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$ *et* $\underline{\mathbf{h}} = \underline{\mathbf{h}}^*$*), à condition que les vecteurs* $\underline{\mathbf{u}}(t=0)$ *et* $\underline{\mathbf{u}}^*$ *forment un angle aigu.*

Preuve :

La convergence vers $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$ est assurée par le Lemme 1. Qu'en est-il pour la convergence vers $\underline{\mathbf{h}} = \underline{\mathbf{h}}^*$?

La commande $\boldsymbol{\Omega} = \mu_u \underline{\mathbf{u}} \times \underline{\mathbf{u}}^*$, $\mu_u > 0$ engendre la variation de la droite image selon :

$$\dot{\underline{\mathbf{h}}} = \mu_u (\underline{\mathbf{u}} \times \underline{\mathbf{u}}^*) \times \underline{\mathbf{h}}^* = -\mu_u \underline{\mathbf{h}}^T \underline{\mathbf{u}}^* \underline{\mathbf{u}}$$

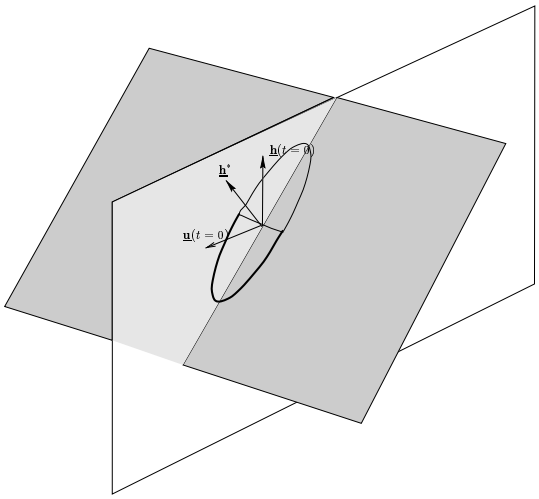


FIG. 2.2: *Selon le Lemme 2, la commande en rotation converge lorsque* $\underline{\mathbf{u}}^*$ *se trouve sur l'arc de cercle en trait plein.*

Première conséquence : la variation de $\underline{\mathbf{h}}$ se fait selon $\underline{\mathbf{u}}$ donc dans le plan $(\underline{\mathbf{u}}, \underline{\mathbf{h}}^*)$ et les vecteurs $\underline{\mathbf{u}}$, $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ restent coplanaires.

Reportons ce résultat dans la dérivée de la fonction de Lyapunov $L_h = \frac{1}{2}\|\underline{\mathbf{h}} - \underline{\mathbf{h}}^*\|^2$:

$$\dot{L}_h = \mu_u \underline{\mathbf{h}}^T \underline{\mathbf{u}}^* \underline{\mathbf{u}}^T \underline{\mathbf{h}}^*$$

que nous allons chercher à simplifier en remplaçant $\underline{\mathbf{h}}^T \underline{\mathbf{u}}^*$.

Comme $\underline{\mathbf{u}}$, $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ sont coplanaires, $\underline{\mathbf{h}}^*$ est perpendiculaire à $\underline{\mathbf{u}} \times \underline{\mathbf{h}}$. On peut donc construire une base orthonormée avec les vecteurs $\underline{\mathbf{h}}^*$, $\underline{\mathbf{u}} \times \underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^* \times (\underline{\mathbf{u}} \times \underline{\mathbf{h}})$. Le vecteur $\underline{\mathbf{u}}^*$ étant orthogonal à $\underline{\mathbf{h}}^*$, il se décompose sur cette base en :

$$\underline{\mathbf{u}}^* = \alpha(\underline{\mathbf{u}} \times \underline{\mathbf{h}}) + \beta \underline{\mathbf{h}}^* \times (\underline{\mathbf{u}} \times \underline{\mathbf{h}})$$

Alors, le produit scalaire $\underline{\mathbf{h}}^T \underline{\mathbf{u}}^*$ devient :

$$\underline{\mathbf{h}}^T \underline{\mathbf{u}}^* = -\beta \underline{\mathbf{u}}^T \underline{\mathbf{h}}^*$$

d'où $\dot{L}_h = -\mu_u (\underline{\mathbf{u}}^T \underline{\mathbf{h}}^*)^2 \beta$. On a donc stabilité asymptotique lorsque $\underline{\mathbf{u}}^T \underline{\mathbf{h}}^* \neq 0$ et $\beta > 0$.

Or, si la première condition n'est pas vérifiée, on a deux contraintes à satisfaire :

$$\begin{aligned} \underline{\mathbf{u}}^T \underline{\mathbf{h}}^* &= 0 \\ (\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T \underline{\mathbf{h}}^* &= 0 \end{aligned}$$

Asservissement visuel à partir de droites

Par conséquent, $\underline{\mathbf{h}} = \pm\underline{\mathbf{h}}^*$. L'angle entre $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ étant, par hypothèse, aigu, on a donc $\underline{\mathbf{h}} = \underline{\mathbf{h}}^*$.

Voyons donc l'autre cas où la stabilité asymptotique n'est pas garantie : $\beta \leq 0$. Pour cela, étudions de plus près la valeur de ce coefficient β. Elle est nulle lorsque $\underline{\mathbf{u}}^* = \pm\underline{\mathbf{u}} \times \underline{\mathbf{h}}$, ce qui est incompatible avec la non orthogonalité de $\underline{\mathbf{h}}$ et $\underline{\mathbf{u}}^*$.

Pour étudier le cas restant où $\beta < 0$, considérons le produit scalaire $\underline{\mathbf{u}}^T\underline{\mathbf{u}}^*$:

$$\underline{\mathbf{u}}^T\underline{\mathbf{u}}^* = \beta\underline{\mathbf{u}}^T[\underline{\mathbf{h}}^* \times (\underline{\mathbf{u}} \times \underline{\mathbf{h}})] = \beta\underline{\mathbf{h}}^T\underline{\mathbf{h}}^*$$

Or, comme l'angle entre $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ est par hypothèse aigu, le produit scalaire $\underline{\mathbf{h}}^T\underline{\mathbf{h}}^*$ est positif. Donc β est de même signe que $\underline{\mathbf{u}}^T\underline{\mathbf{u}}^*$. Ainsi, la stabilité asymptotique n'est pas garantie lorsque $\underline{\mathbf{u}}^T\underline{\mathbf{u}}^* < 0$. □

Commande en translation — La commande en translation est plus ardue à définir. Plaçons-nous provisoirement dans le cas idéal où la commande en rotation a convergé (c.-à-d. lorsque $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$). Alors, $\boldsymbol{\Omega} = 0$ et les équations du mouvement deviennent :

$$\begin{aligned}
\dot{\underline{\mathbf{u}}} &= 0 \\
\dot{\underline{\mathbf{h}}} &= \frac{1}{h}(\mathbf{I}_3 - \underline{\mathbf{h}}\underline{\mathbf{h}}^T)(\mathbf{V} \times \underline{\mathbf{u}}) \\
&= -\frac{1}{h}\mathbf{V}^T\underline{\mathbf{h}}\ (\underline{\mathbf{u}} \times \underline{\mathbf{h}})
\end{aligned}$$

Il faut donc maintenant déterminer l'erreur à réguler à 0 qui permette d'aligner $\underline{\mathbf{h}}$ sur $\underline{\mathbf{h}}^*$. On est tenté d'utiliser une erreur contenant des termes de la forme $\underline{\mathbf{h}} \times \underline{\mathbf{h}}^*$ par similarité avec ce que l'on a fait pour $\underline{\mathbf{u}}$, car $\underline{\mathbf{h}}$ est un vecteur unitaire. Toutefois, c'est une mauvaise idée car cela n'engendre aucune variation de $\underline{\mathbf{h}}$. En effet, si l'on suppose $\boldsymbol{\Omega} = 0$ et si l'on choisit $\mathbf{V} = -\mu_h(\underline{\mathbf{h}} \times \underline{\mathbf{h}}^*)$, l'équation précédente devient :

$$\begin{aligned}
\dot{\underline{\mathbf{h}}} &= -\frac{1}{h}\mathbf{V}^T\underline{\mathbf{h}}\ (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \\
&= -\frac{1}{h}(-\mu_h)\underbrace{(\underline{\mathbf{h}} \times \underline{\mathbf{h}}^*)^T\underline{\mathbf{h}}}_{0}\ (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \\
&= 0
\end{aligned}$$

Le déplacement se fait donc dans le plan d'interprétation de (D). On peut d'ailleurs le voir d'une seconde manière. En effet, $\mathbf{V}$ est proportionnelle au produit vectoriel de $\underline{\mathbf{h}}$ par $\underline{\mathbf{h}}^*$. Elle est donc perpendiculaire à $\underline{\mathbf{h}}$ et, par conséquent,

dans le plan d'interprétation. De plus, on remarquera que dans le cas où $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$, le déplacement demandé est en fait une translation le long de la droite (D).

On ne peut pas non plus choisir une erreur proportionnelle à la différence $\underline{\mathbf{h}} - \underline{\mathbf{h}}^*$. Vérifions-le en insérant $\boldsymbol{\Omega} = 0$ et $\mathbf{V} = -\mu_h(\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)$ dans (2.7) :

$$\begin{aligned} \dot{\underline{\mathbf{h}}} &= -\frac{1}{h}(-\mu_h)(\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)^T \underline{\mathbf{h}} \; (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \\ &= \frac{\mu_h}{h}(1 - \underline{\mathbf{h}}^T \underline{\mathbf{h}}^*) \; (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \end{aligned}$$

Ce choix de vitesse fait donc tourner $\underline{\mathbf{h}}$ autour de $\underline{\mathbf{u}}$ jusqu'à ce que $\underline{\mathbf{h}} = \underline{\mathbf{h}}^*$. Avec l'hypothèse en cours ($\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$), cette condition peut être remplie. Toutefois, il en va autrement avant la convergence de $\underline{\mathbf{u}}$ vers $\underline{\mathbf{u}}^*$. En effet, $\underline{\mathbf{h}}$ est contraint à être orthogonal à $\underline{\mathbf{u}}$ mais cette contrainte ne s'applique pas à $\underline{\mathbf{h}}^*$. Dès lors, il est possible que $\underline{\mathbf{h}}$ ne rejoigne jamais $\underline{\mathbf{h}}^*$, entraînant la caméra dans un mouvement globalement tournant autour de la droite (D).

On peut alors envisager de réduire une erreur de la forme : $\underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)$. Vérifions-le en insérant $\boldsymbol{\Omega} = 0$ et $\mathbf{V} = -\mu_h \, \underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)$ (avec $\mu_h > 0$) dans (2.7) :

$$\begin{aligned} \dot{\underline{\mathbf{h}}} &= -\frac{1}{h}(-\mu_h) \left[\underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)\right]^T \underline{\mathbf{h}} \; (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \\ &= \frac{\mu_h}{h} \left[\underbrace{(\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T \underline{\mathbf{h}}}_{0} - (\underline{\mathbf{u}} \times \underline{\mathbf{h}}^*)^T \underline{\mathbf{h}} \right] \; (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \end{aligned}$$

On obtient donc finalement :

$$\dot{\underline{\mathbf{h}}} = \frac{\mu_h}{h} \left[(\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T \underline{\mathbf{h}}^*\right] \; (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \tag{2.10}$$

Cette commande n'a donc plus d'influence sur $\underline{\mathbf{h}}$ lorsque le produit mixte $(\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T \underline{\mathbf{h}}^*$ s'annule, c'est-à-dire lorsque $\underline{\mathbf{u}}$, $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ sont coplanaires. On notera que l'on retrouve ici la première prémisse du Théorème 2.

Ce produit mixte est directement relié à la distance naturelle entre les vecteurs unitaires $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ puisqu'il est aussi égal à $(\underline{\mathbf{h}} \times \underline{\mathbf{h}}^*)^T \underline{\mathbf{u}}$. Néanmoins, cette dernière expression laisse entrevoir la présence de singularités que nous étudierons immédiatement après avoir terminé l'interprétation géométrique de cette commande.

Le choix que l'on a fait pour $\mathbf{V}$ a pour effet de faire tourner $\underline{\mathbf{h}}$ perpendiculairement à $\underline{\mathbf{u}}$ jusqu'à ce qu'il se trouve dans le plan $(\underline{\mathbf{u}}, \underline{\mathbf{h}}^*)$ (voir Figure 2.3). On notera que le produit mixte oriente $\dot{\underline{\mathbf{h}}}$ dans le sens du plus court chemin vers $\underline{\mathbf{h}}^*$ (Figure 2.4) pour aboutir à la position minimisant l'écart entre $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$. Il existe cependant deux cas singuliers.

Le premier apparaît lorsque $\underline{\mathbf{u}} = \underline{\mathbf{h}}^*$, c.-à-d. lorsque la droite est, à l'instant courant, perpendiculaire au plan d'interprétation désiré. Cependant, une faible perturbation de $\underline{\mathbf{u}}$ permet de sortir de cette singularité et nous ramène dans le cas général de la Figure 2.4.

Asservissement visuel à partir de droites

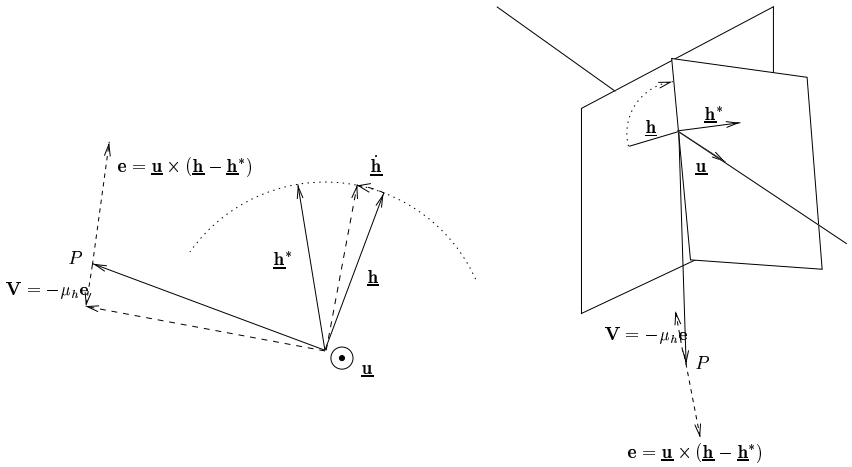


FIG. 2.3: *L'effet de la vitesse* $\mathbf{V} = -\mu_h \underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)$ *: à gauche, dans le cas où* $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$ *et à droite, dans le cas contraire.*

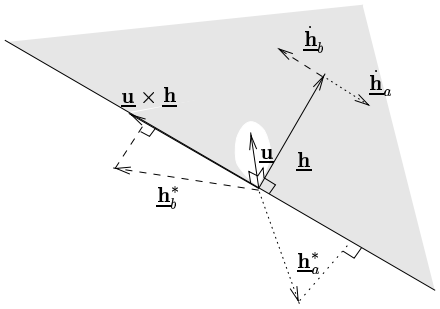


FIG. 2.4: *La vitesse* $\mathbf{V} = -\mu_h \underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)$ *oriente* $\dot{\underline{\mathbf{h}}}$ *dans le sens du plus court chemin vers* $\underline{\mathbf{h}}^*$

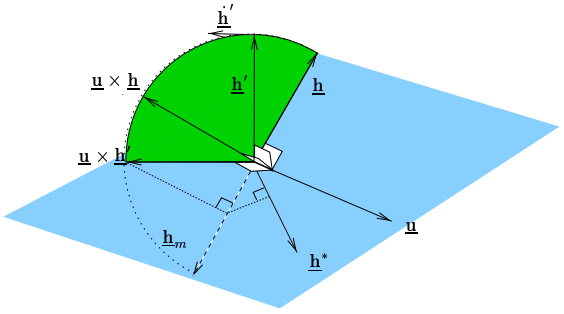


FIG. 2.5: *Dans le cas singulier où* $\dot{\underline{\mathbf{h}}} = 0$ *sans que* $\underline{\mathbf{h}}$ *et* $\underline{\mathbf{h}}^*$ *aient un écart minimal, une perturbation* $\underline{\mathbf{h}}'$ *de* $\underline{\mathbf{h}}$ *génère une variation* $\dot{\underline{\mathbf{h}}}'$ *non-nulle qui ramène* $\underline{\mathbf{h}}$ *vers la position* $\underline{\mathbf{h}}_m$ *de moindre écart*

Le second cas singulier intervient lorsque $\underline{\mathbf{h}}^*$ est initialement dans le plan $(\underline{\mathbf{u}}, \underline{\mathbf{h}})$, formant un angle obtus avec $\underline{\mathbf{h}}$ (Figure 2.5). Alors, $\dot{\underline{\mathbf{h}}} = 0$ mais l'on n'a pas l'écart minimal. Cependant, on peut remarquer qu'une faible perturbation du vecteur $\underline{\mathbf{h}}$ permet de le ramener vers la position d'écart minimal. En pratique, ce cas ne devrait donc pas persister.

Tout ceci se résume avec le lemme suivant :

Lemme 3

Soit une droite (D) *de coordonnées courantes* $(\underline{\mathbf{h}}, \underline{\mathbf{u}}, h)$ *que l'on souhaite amener en une position réalisant l'alignement en coordonnées de Plücker binormées suivant :*

$$\begin{aligned} \underline{\mathbf{h}} &= \underline{\mathbf{h}}^* \\ \underline{\mathbf{u}} &= \underline{\mathbf{u}}^* \end{aligned}$$

où la notation * *est la marque de la valeur souhaitée.*
Alors, le choix de la fonction de tâche

$$\mathbf{e}_h = \underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)$$

et de la commande

$$\begin{aligned} \mathbf{\Omega} &= 0 \\ \mathbf{V} &= -\mu_h \, \mathbf{e}_h \ (\mu_h > 0) \end{aligned}$$

minimise, à $\underline{\mathbf{u}}$ *constant (et non forcément égal à* $\underline{\mathbf{u}}^*$*) la distance* $(\underline{\mathbf{h}} \times \underline{\mathbf{h}}^*)$ *entre* $\underline{\mathbf{h}}$ *et* $\underline{\mathbf{h}}^*$.
Les singularités, définies par $(\underline{\mathbf{u}} = \underline{\mathbf{h}}^*)$ *et* $((\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T \underline{\mathbf{h}}^* = 0$ et $\underline{\mathbf{h}} \times \underline{\mathbf{h}}^*$non minimale), *sont des équilibres instables.*

En pratique, si la commande en rotation est activée avant celle en translation, alors $\underline{\mathbf{u}}$ est soit égal à $\underline{\mathbf{u}}^*$ (et donc forcément différent de $\underline{\mathbf{h}}^*$), soit attiré vers cette valeur (et donc repoussé par $\underline{\mathbf{h}}^*$). La singularité $(\underline{\mathbf{u}} = \underline{\mathbf{h}}^*)$ pourra donc être traversée sans dommage.

En supposant que la commande en translation est activée après la convergence de la commande en rotation, on obtient le théorème de convergence suivant :

Lemme 4 (Lemme de convergence)

Lorsque $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$*, et dans le cas idéal où la commande en translation définie au Lemme 3 est parfaite (c.-à-d. n'engendre pas de rotations parasites), alors cette dernière converge asymptotiquement vers l'alignement en coordonnées de Plücker binormées si la condition initiale* $\underline{\mathbf{h}}(t=0) \neq -\underline{\mathbf{h}}^*$ *est vérifiée.*
Cette condition représente un équilibre instable.

Asservissement visuel à partir de droites

Preuve :

Considérons la fonction de Lyapunov suivante :

$$L_h = \frac{1}{2}\|\underline{\mathbf{h}} - \underline{\mathbf{h}}^*\|^2$$

La dérivée $\dot{L}_h$ de cette fonction s'écrit :

$$\dot{L}_h = \dot{\underline{\mathbf{h}}}^T(\underline{\mathbf{h}} - \underline{\mathbf{h}}^*)$$

et se simplifie en :

$$\dot{L}_h = -\dot{\underline{\mathbf{h}}}^T\underline{\mathbf{h}}^* \tag{2.11}$$

puisque $\underline{\mathbf{h}}$ est unitaire.

En remplaçant $\dot{\underline{\mathbf{h}}}$ par l'expression obtenue en (2.10), on obtient :

$$\dot{L}_h = -\frac{\mu_h}{h}\left[(\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T\underline{\mathbf{h}}^*\right]\ (\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T\underline{\mathbf{h}}^*$$

qui se factorise en :

$$\dot{L}_h = -\frac{\mu_h}{h}\left[(\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T\underline{\mathbf{h}}^*\right]^2$$

On voit donc clairement que cette dérivée est négative lorsque μ_h est strictement positif et qu'elle s'annule lorsque $\underline{\mathbf{u}}$, $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ sont coplanaires.

Cependant, lorsque $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$ (convergence de la commande du Lemme 1), cette clause de coplanarité se ramène à une clause de colinéarité de $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$. En effet, $\underline{\mathbf{h}}$ est alors contraint à être l'intersection du plan perpendiculaire à $\underline{\mathbf{u}}^*$ et du plan vectoriel défini par $\underline{\mathbf{u}}^*$ et $\underline{\mathbf{h}}^*$, donc à appartenir à la droite vectorielle définie par $\underline{\mathbf{h}}^*$.

Comme $\underline{\mathbf{h}}$ est unitaire, $\dot{L}_h$ s'annule donc en 2 cas : lorsque $\underline{\mathbf{h}} = \underline{\mathbf{h}}^*$ et lorsque $\underline{\mathbf{h}} = -\underline{\mathbf{h}}^*$. Le premier cas correspond à la convergence et le second au deuxième des équilibres instables présentés au Lemme 3. □

Commande complète — Il s'agit désormais de regrouper la commande en rotation et celle en translation. Il existe pour cela plusieurs stratégies possibles.

- Comme la commande en rotation est inconditionnellement convergente (Lemme 1) et que la commande en translation converge lorsque $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$ (Lemme 3), on peut envisager une commande en deux phases: commande en rotation puis, à convergence, commande en translation.

Cependant, à cause des imperfections du monde réel, la commande en translation génère des rotations résiduelles. Celle-ci engendre alors des variations de la direction des droites par rapport à leur direction désirée. Ces variations sont probablement de faible amplitude mais risquent de s'accumuler et de faire diverger la commande. Il faut donc garder active la commande en rotation pour absorber ces erreurs résiduelles. On obtient donc ainsi la commande complète en deux phases successives :

$$\mathbf{e}_u = \underline{\mathbf{u}} \times \underline{\mathbf{u}}^* \tag{2.12}$$

$$\boldsymbol{\Omega} = \mu_u \, \mathbf{e}_u, \quad \mu_u > 0 \tag{2.13}$$

$$\mathbf{e}_h = \underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*) \tag{2.14}$$

$$\mathbf{V} = -\mu_h \, \mathbf{e}_h, \quad \begin{cases} \mu_h = 0 & si \ \|\mathbf{e}_u\| \leq \epsilon \\ \mu_h > 0 & sinon \end{cases} \tag{2.15}$$

La transition entre les deux phases de la commande se fait ci-dessus par commutation. Pour obtenir une transition continue, donc moins brutale, on peut remplacer μ_h par une fonction continue de $\|\mathbf{e}_u\|$, maximale lorsque $\mathbf{e}_u = 0$ et décroissant rapidement lorsque $\|\mathbf{e}_u\|$ augmente, par exemple :

$$\mu(\|\mathbf{e}_u\|) = \mu_h e^{-k\|\mathbf{e}_u\|}, k >> 0$$

Cette commande en deux phases est acceptable lorsque la caméra est déportée. En revanche, un problème peut néanmoins surgir, dans le cas d'une caméra embarquée, si le mouvement séparant la position initiale de la position désirée de la droite comporte une rotation de grande amplitude (voir Figure 2.6). En effet, la commande en rotation aura pour effet de faire tourner la caméra sur elle-même, ce qui peut faire sortir la droite du champ de vue de la caméra. Pourtant, il existe de nombreuses trajectoires qui permettent de ne pas perdre la droite de vue !

Pour éviter ce problème, on peut ajouter dans la première phase une commande empirique en translation chargée de conserver la droite dans l'image, par exemple une commande qui minimise la variation de la distance du centre de l'image à l'image de la droite ;

– En utilisant plutôt les Lemmes 2 et 3, on peut utiliser la stratégie inverse : commande en translation puis commande en rotation.

La première permet, d'après le Lemme 3, d'obtenir les prémisses du Lemme 2 qui garantit la convergence de la deuxième commande vers l'alignement en coordonnées de Plücker binormées ($\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$ et $\underline{\mathbf{h}} = \underline{\mathbf{h}}^*$).

Asservissement visuel à partir de droites

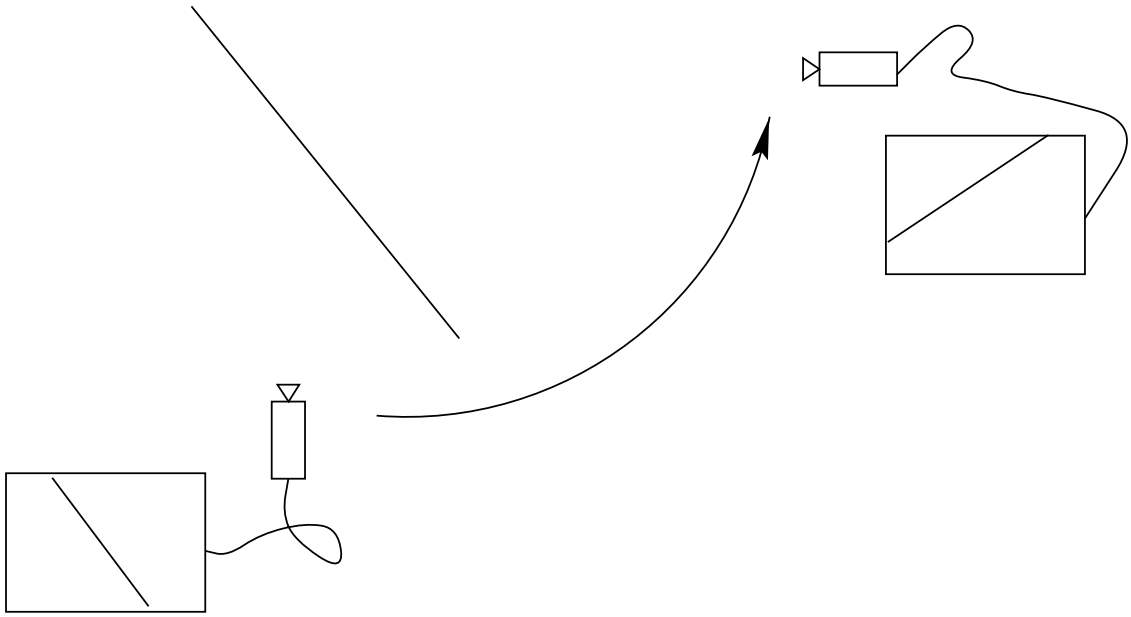


FIG. 2.6: *Cette configuration où une rotation de* 90° *sépare la position initiale de la position désirée peut faire échouer la commande en deux phases.*

Ici aussi, on est amené en pratique à garder active la commande de la première phase après la commutation vers la deuxième pour compenser l'imperfection du monde physique. De même, on peut remplacer la commutation par une transition plus douce telle que nous l'avons vue dans la stratégie précédente.

Cependant, la convergence de la deuxième phase n'est garantie que si les vecteurs $\underline{\mathbf{u}}$ et $\underline{\mathbf{u}}^*$ forment initialement un angle aigu et il est souhaitable de trouver une solution plus générale ;

- On peut alors envisager une stratégie alternant les deux commandes. Ainsi, on active dans un premier temps, la commande en rotation qui nous offre la garantie de réduire l'erreur en orientation. Puis, soit à sa convergence vers $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$, soit lorsque la droite est trop proche de sortir de l'image, on commute vers la commande en translation aussi longtemps que nécessaire pour minimiser l'angle entre $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ (c.-à-d. ramener la droite image le plus près possible de sa position souhaitée). On revient alors vers la commande en rotation qui va soit converger vers $\underline{\mathbf{u}} = \underline{\mathbf{u}}^*$ et $\underline{\mathbf{h}} = \underline{\mathbf{h}}^*$, soit réduire l'angle entre $\underline{\mathbf{u}}$ et $\underline{\mathbf{u}}^*$ et augmenter l'écart entre $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$, et ainsi de suite. D'après les théorèmes énoncés jusqu'à présent dans ce chapitre, on doit très certainement obtenir la convergence.

- On peut enfin activer simultanément les deux phases. Il reste dans ce cas à vérifier, à partir de la théorie des systèmes cascadés [PL98], si la commande

conserve ses propriétés de stabilité. Cela semble probable puisque cette dernière stratégie est un passage à la limite de la précédente et que, de plus, le Jacobien associé aux coordonnées de Plücker binormées possède une structure triangulaire par bloc.

Plusieurs droites

Dans le paragraphe qui précède, nous avons mis à jour une loi de commande inversant analytiquement les équations du mouvement d'une droite. L'intérêt de cette solution analytique est qu'elle s'étend au cas où plusieurs droites sont observées dans la scène. Pour cela, introduisons à nouveau l'indice i ($i = 1..n$) de chaque droite.

Pour pouvoir bénéficier des résultats obtenus pour une droite, on cherche une commande en deux phases combinant les influences de chaque droite :

$$\mathbf{e}_{u_i} = \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^* \tag{2.16}$$

$$\mathbf{\Omega} = \mu_u \, \mathbf{C}_u \begin{pmatrix} \mathbf{e}_{u_1} \\ \vdots \\ \mathbf{e}_{u_n} \end{pmatrix}, \quad \mu_u > 0 \tag{2.17}$$

$$\mathbf{e}_{h_i} = \underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*) \tag{2.18}$$

$$\mathbf{V} = -\mu_h \, \mathbf{C}_h \begin{pmatrix} \mathbf{e}_{h_1} \\ \vdots \\ \mathbf{e}_{h_n} \end{pmatrix}, \quad \begin{cases} \mu_h = 0 & si \ \|\mathbf{e}_u\| \leq \epsilon \\ \mu_h > 0 & sinon \end{cases} \tag{2.19}$$

où $\mathbf{C}_u$ et $\mathbf{C}_h$ sont des matrices $(3 \times 3n)$ de rang plein qu'il nous faut caractériser. Ces matrices de combinaison doivent être telles qu'elles permettent la stabilité asymptotique de la loi de commande ainsi définie.

Première étape : rotation — Considérons d'abord la première phase de commande en rotation uniquement. Alors nous avons la proposition suivante :

Proposition 1

La commande en rotation seulement, première phase de la commande proposée en (2.16)–(2.19), est bornée si le produit $\mathbf{UC}_u$ *est une matrice symétrique semi-définie positive, où* $\mathbf{U}$ *est la matrice* $(3n \times 3)$ *définie par :*

$$\mathbf{U} = \begin{pmatrix} \mathbf{I}_3 \\ \vdots \\ \mathbf{I}_3 \end{pmatrix}$$

Asservissement visuel à partir de droites

Preuve :

Considérons la fonction de Lyapunov suivante :

$$L_u = \frac{1}{2}\sum_{i=1}^{n} \|\underline{\mathbf{u}}_i - \underline{\mathbf{u}}_i^*\|^2$$

extension de celle utilisée dans le Lemme 1. Sa dérivée est alors :

$$\dot{L}_u = -\sum_{i=1}^{n} \dot{\underline{\mathbf{u}}}_i^T \underline{\mathbf{u}}_i^* \tag{2.20}$$

Comme dans le cas d'une seule droite, la seule équation du mouvement qui nous intéresse est (1.16) qui s'écrit :

$$\dot{\underline{\mathbf{u}}}_i = (\mu_u \mathbf{C}_u \mathbf{E}_u) \times \underline{\mathbf{u}}_i, \quad i = 1..n$$

avec la notation abrégée :

$$\mathbf{E}_u = \begin{pmatrix} \mathbf{e}_{u_1} \\ \vdots \\ \mathbf{e}_{u_n} \end{pmatrix} = \begin{pmatrix} \underline{\mathbf{u}}_1 \times \underline{\mathbf{u}}_1^* \\ \vdots \\ \underline{\mathbf{u}}_n \times \underline{\mathbf{u}}_n^* \end{pmatrix}$$

En introduisant la notation matricielle du produit vectoriel, l'expression précédente se réécrit sous la forme suivante :

$$\dot{\underline{\mathbf{u}}}_i = -\mu_u As(\underline{\mathbf{u}}_i)\mathbf{C}_u\mathbf{E}_u$$

En reportant cette expression dans (2.20), on obtient :

$$\dot{L}_u = \mu_u \sum_{i=1}^{n} \underline{\mathbf{u}}_i^{*T}(As(\underline{\mathbf{u}}_i)\mathbf{C}_u\mathbf{E}_u)$$

On remarque que $\underline{\mathbf{u}}_i^{*T} As(\underline{\mathbf{u}}_i) = -(\underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*)^T$. En reportant ce résultat dans l'expression ci-dessus dans laquelle on peut factoriser le terme $\mathbf{C}_u\mathbf{E}_u$, on aboutit à :

$$\dot{L}_u = -\mu_u \left(\sum_{i=1}^{n} (\underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*)^T\right) \mathbf{C}_u\mathbf{E}_u$$

On peut voir que $\sum_{i=1}^{n}(\underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*)^T$ s'écrit matriciellement comme $\mathbf{E}_u^T\mathbf{U}$, d'où l'expression finale de $\dot{L}_u$:

$$\dot{L}_u = -\mu_u \mathbf{E}_u^T \mathbf{U} \mathbf{C}_u \mathbf{E}_u$$

Le produit $\mathbf{UC}_u$ ne peut être de rang plein car il est égal à la matrice bloc :

$$\begin{pmatrix} \mathbf{C}_u \\ \vdots \\ \mathbf{C}_u \end{pmatrix}$$

où, rappelons-le, la matrice $\mathbf{C}_u$ est de dimension $(3\times 3n)$. Par conséquent, il ne peut être au maximum que semi défini positif.

Si ce produit n'est que semi défini positif, alors $\dot{L}_u$ est négative, mais peut s'annuler même lorsque $\mathbf{E}_u \neq 0$ (voir [Kha96, p.115] pour plus de précision). Les erreurs $\|\underline{\mathbf{u}}_i - \underline{\mathbf{u}}_i^*\|$ $(i = 1..n)$ (ou, ce qui est équivalent, le vecteur $\mathbf{E}_u$) sont donc non strictement décroissantes et sont stationnaires lorsque $\dot{L}_u = 0$.

On notera que le fait que $\mathbf{E}_u$ soit nul équivaut à ce que les $\underline{\mathbf{u}}_i$ soient ou bien égaux ou bien opposés aux $\underline{\mathbf{u}}_i^*$. On retrouve donc les mêmes singularités initiales instables que dans le cas d'une droite. □

Un choix naturel pour $\mathbf{C}_u$ est $\mathbf{U}^T$ qui correspond tout simplement à additionner les commandes élémentaires définies par chacune des droites. Ce choix est validé par la proposition suivante :

Proposition 2

On peut garantir la décroissance (non stricte) des erreurs $\|\underline{\mathbf{u}}_i - \underline{\mathbf{u}}_i^*\|$ *pour* $i = 1..n$ *en choisissant* $\mathbf{C}_u = \mathbf{U}^T$. *Les états stationnaires du système sont alors tels que:* $\sum_{i=1}^{n} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^* = 0$.

Preuve :

On peut vérifier que le produit $\mathbf{UU}^T$ est semi défini positif. Par exemple, on peut l'exprimer à l'aide du produit de Kronecker (voir Chapitre 1 de la deuxième partie) et utiliser les propriétés de ce dernier [Bre78].

En se reportant à la preuve précédente, on constate qu'en plus des conditions initiales instables, la dérivée $\dot{L}_u$ s'annulera lorsque $\mathbf{U}^T\mathbf{E}_u$ le fera. Comme cette dernière expression est égale à $\sum_{i=1}^{n} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*$, on a la caractérisation des états stationnaires. □

Asservissement visuel à partir de droites

Cette proposition est inquiétante puisqu'il semble exister une kyrielle d'états stationnaires. En fait, nous sommes sauvés par le lemme suivant :

Lemme 5

Soient $\underline{\mathbf{u}}_i$, $(i = 1..n)$, *les directions à l'instant courant de* n *droites* (D_i) *rigidement liées et* $\underline{\mathbf{u}}_i^*$, *les directions désirées de ces mêmes droites. Alors,*

$$\sum_{i=1}^{n} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^* = 0 \Rightarrow \underline{\mathbf{u}}_i = \pm \underline{\mathbf{u}}_i^*, i = 1..n$$

Preuve :

Cette preuve se base sur la formule de Rodrigues qui relie un vecteur $\mathbf{v}$ à son image $\mathbf{v}'$ par une rotation d'axe $\underline{\mathbf{n}}$ et d'angle θ par :

$$\mathbf{v}' = \mathbf{v} + \sin\theta(\underline{\mathbf{n}} \times \mathbf{v}) + (1 - \cos\theta)\underline{\mathbf{n}} \times (\underline{\mathbf{n}} \times \mathbf{v})$$

Comme les (D_i) sont rigidement liées, les $\underline{\mathbf{u}}_i$ sont séparés des $\underline{\mathbf{u}}_i^*$ par une même rotation. Par conséquent, on a :

$$\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_i^* + \sin\theta(\underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*) + (1 - \cos\theta)\underline{\mathbf{n}} \times (\underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*), \quad i = 1..n$$

grâce à quoi, nous obtenons :

$$\underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^* = \sin\theta(\underline{\mathbf{n}}^T \underline{\mathbf{u}}_i^* \underline{\mathbf{u}}_i^* - \underline{\mathbf{n}}) + (1 - \cos\theta)\underline{\mathbf{n}}^T \underline{\mathbf{u}}_i^* (\underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*), \quad i = 1..n$$

Or, chaque $\underline{\mathbf{u}}_i^*$ se décompose respectivement sur la base orthonormée directe $(\underline{\mathbf{n}}, \underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*, \underline{\mathbf{n}} \times (\underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*))$ en :

$$\underline{\mathbf{u}}_i^* = \underline{\mathbf{n}}^T \underline{\mathbf{u}}_i^* \underline{\mathbf{n}} + \alpha_i(\underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*) + \beta_i \underline{\mathbf{n}} \times (\underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*), \quad i = 1..n$$

Par conséquent, les produits vectoriels $\underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*$ deviennent :

$$\begin{aligned} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^* &= \sin\theta\Big[(\underline{\mathbf{n}}^T \underline{\mathbf{u}}_i^*)^2 - 1\Big]\underline{\mathbf{n}} \\ &+ \Big[(1 - \cos\theta)\underline{\mathbf{n}}^T \underline{\mathbf{u}}_i^* + \alpha_i \underline{\mathbf{n}}^T \underline{\mathbf{u}}_i^*\Big] \underbrace{(\underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*)}_{\perp \underline{\mathbf{n}}} \\ &+ \Big[\beta_i \underline{\mathbf{n}}^T \underline{\mathbf{u}}_i^*\Big] \underbrace{\underline{\mathbf{n}} \times (\underline{\mathbf{n}} \times \underline{\mathbf{u}}_i^*)}_{\perp \underline{\mathbf{n}}} \end{aligned}$$

Ainsi, si la somme de ces produits vectoriels est nulle, alors :

$$\sum_{i=1}^{n} \sin\theta\Big[(\underline{\mathbf{n}}^T \underline{\mathbf{u}}_i^*)^2 - 1\Big] = 0$$

Ceci implique, hors le cas où $\sin\theta = 0$ qui signifie que les $\underline{\mathbf{u}}_i$ sont égaux ou opposés aux $\underline{\mathbf{u}}_i^*$, que

$$\sum_{i=1}^{n}[(\underline{\mathbf{n}}^T\underline{\mathbf{u}}_i^*)^2 - 1] = 0$$

Cependant, chaque terme $(\underline{\mathbf{n}}^T\underline{\mathbf{u}}_i^*)^2 - 1$ est négatif, car les vecteurs qui y figurent sont unitaires. Par conséquent, la somme ci-dessus n'est nulle que si :

$$(\underline{\mathbf{n}}^T\underline{\mathbf{u}}_i^*)^2 - 1 = 0, \quad i = 1..n$$

d'où, $\underline{\mathbf{u}}_i^* = \pm\underline{\mathbf{n}}$ pour tout i et, en reportant dans la formule de Rodrigues $\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_i^*$. □

Conséquence finale de ce lemme, la commande est bornée, possède les états stationnaires tels que $\underline{\mathbf{u}}_i = \pm\underline{\mathbf{u}}_i^*$ pour tout i et ne peut atteindre ceux qui comportent des indices j tels que $\underline{\mathbf{u}}_j = -\underline{\mathbf{u}}_j^*$. On obtient donc finalement le théorème de convergence suivant :

Théorème 1 (Théorème de convergence)

La commande en rotation

$$\boldsymbol{\Omega} = \mu_u \sum_{i=1}^{n} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*, \quad \mu_u > 0$$

converge asymptotiquement vers

$$\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_i^*, \quad i = 1..n$$

à moins que le système ne soit initialement dans un des états stationnaires (instables, cependant) définis par :

$$\exists\ j \in \{1, .., n\},\ \underline{\mathbf{u}}_j(t=0) = -\underline{\mathbf{u}}_j^*.$$

Ce théorème étend donc à un nombre quelconques de droites les résultats du Lemme 1, qui garantit la convergence asymptotique de la direction d'une droite unique vers sa valeur de consigne. En revanche, le Lemme 2, qui, sous certaines conditions initiales, garantit la convergence asymptotique vers l'alignement en coordonnées de Plücker binormées, n'a pas encore pu être étendu.

Deuxième étape : translation — Comme dans le cas d'une seule droite, nous nous plaçons dans le cas idéal où la commande en translation n'engendre pas de rotation parasite, ce qui nous permet de faire l'hypothèse $\mathbf{\Omega} = 0$. On peut alors énoncer le théorème suivant :

Théorème 2 (Condition suffisante de stabilité)

Grâce à la commande en translation, définie par :

$$\begin{aligned} \mathbf{e}_{h_i} &= \underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*) \\ \mathbf{V} &= -\mu_h \mathbf{C}_h \begin{pmatrix} \mathbf{e}_{h_1} \\ \vdots \\ \mathbf{e}_{h_n} \end{pmatrix}, \quad \mu_h > 0 \end{aligned}$$

et deuxième phase de la commande proposée en (2.16)–(2.19), les erreurs $\mathbf{e}_{h_i}$ $(i = 1..n)$ sont non strictement décroissantes si le produit $\mathbf{HC}_h$ est une matrice symétrique semi définie positive où $\mathbf{H}$ est la matrice $(3n \times 3)$ définie par :

$$\mathbf{H} = \begin{pmatrix} \frac{\underline{\mathbf{h}}_1 \underline{\mathbf{h}}_1^T}{h_1} \\ \vdots \\ \frac{\underline{\mathbf{h}}_n \underline{\mathbf{h}}_n^T}{h_n} \end{pmatrix}$$

Il n'est pas possible de trouver une matrice telle que $\mathbf{HC}_h$ soit définie positive (condition de décroissance stricte).

Preuve :

Considérons la fonction de Lyapunov suivante :

$$L_h = \frac{1}{2} \sum_{i=1}^{n} \mathbf{e}_{h_i}^T \mathbf{e}_{h_i}$$

où l'erreur $\mathbf{e}_{h_i}$ est celle que l'on régule à 0, à savoir $\mathbf{e}_{h_i} = \underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*)$. Alors, la dérivée de L_h est :

$$\dot{L}_h = \sum_{i=1}^{n} (\underline{\mathbf{u}}_i \times \dot{\underline{\mathbf{h}}}_i)^T \mathbf{e}_{h_i}$$

Comme on suppose que la translation est parfaite, on a donc $\mathbf{\Omega} = 0$. Par conséquent la seule équation du mouvement qu'il nous faut prendre en compte est (2.7) qui s'écrit :

$$\dot{\underline{\mathbf{h}}}_i = -\frac{V^T \underline{\mathbf{h}}_i}{h_i} (\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i), \quad i = 1..n$$

Pour simplifier les notations, notons $\mathbf{E}_h = \begin{pmatrix} \mathbf{e}_{h_1} \\ \vdots \\ \mathbf{e}_{h_n} \end{pmatrix}$.

En remplaçant dans l'équation du mouvement $\mathbf{V}$ par $(\mathbf{V} = -\mu_h \mathbf{C}_h \mathbf{E}_h)$, expression donnée par la loi de commande (2.19), on obtient :

$$\dot{\underline{\mathbf{h}}}_i = \frac{\mu_h}{h_i} \mathbf{E}_h^T \mathbf{C}_h^T \underline{\mathbf{h}}_i (\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i), \quad i = 1..n$$

Le produit $(\underline{\mathbf{u}}_i \times \dot{\underline{\mathbf{h}}}_i)$ apparaissant dans l'expression de $\dot{L}_h$ vaut donc :

$$\underline{\mathbf{u}}_i \times \dot{\underline{\mathbf{h}}}_i = -\frac{\mu_h}{h_i} \mathbf{E}_h^T \mathbf{C}_h^T \underline{\mathbf{h}}_i \underline{\mathbf{h}}_i, \quad i = 1..n$$

car $\underline{\mathbf{u}}_i \times (\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i) = -\underline{\mathbf{h}}_i$. En reportant ce résultat dans $\dot{L}_h$, on trouve alors :

$$\dot{L}_h = -\mu_h \sum_{i=1}^{n} \frac{1}{h_i} \mathbf{E}_h^T \mathbf{C}_h^T \underline{\mathbf{h}}_i \underline{\mathbf{h}}_i^T \mathbf{e}_{h_i}$$

qui se réécrit sous forme matricielle comme :

$$\dot{L}_h = -\mu_h \mathbf{E}_h^T \mathbf{C}_h^T \mathbf{H}^T \mathbf{E}_h$$

Par conséquent, si le produit $\mathbf{HC}_h$ est symétrique semi défini positif, alors le système réduit au sens large la norme du vecteur $\mathbf{E}_h$ et donc les $\mathbf{e}_{h_i}$.

Le produit $\mathbf{C}_h^T \mathbf{H}^T$ (ou de manière équivalente, le produit $\mathbf{HC}_h$) ne peut être défini positif car il ne peut pas être de rang plein. En effet, $\mathbf{HC}_h$ est de dimension $(3n \times 3n)$ alors que chacune des deux matrices est au maximum de rang 3. Donc, $\mathbf{HC}_h$ est aussi au maximum de rang 3, loin du rang $3n$ escompté.

□

Il vient alors immédiatement :

Proposition 3

Le choix $\mathbf{C}_h = \mathbf{H}^T$ *vérifie le théorème précédent.*

Contrairement à la commande en rotation pour laquelle nous avons pu choisir une matrice de combinaison $\mathbf{C}_u$ indépendante de l'état du système qui nous permette de tirer des conclusions *a priori* quant à la convergence de la commande, ici, ce n'est plus possible. Nous verrons toutefois, dans le Chapitre 3, que, malgré tout, on peut obtenir quelques résultats de convergence dans la configuration particulière d'un faisceau de 3 droites orthogonales et avec $\mathbf{C}_h = \mathbf{U}^T$.

Asservissement visuel à partir de droites

2.3.4 Commande projetée

La commande CPN projetée conserve la séparation en deux phases de la commande précédente pour garder les propriétés de découplage entre rotation et translation. De plus, la première phase est inchangée puisqu'elle possède des propriétés sympathiques.

En revanche, la seconde phase est modifiée. Nous allons en voir la raison dans le cas d'une droite et la conséquence positive sur la condition de stabilité dans le cas d'un nombre quelconque de droites.

Une droite

Tout part de la remarque que dans la commande précédente, que nous rappelons ici :

$$\begin{aligned} \mathbf{V} &= -\mu_h \underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*), \quad \mu_h > 0 \\ \dot{\underline{\mathbf{h}}} &= \frac{\mu_h}{h} \left[(\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T \underline{\mathbf{h}}^* \right] \ (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \end{aligned}$$

la vitesse de translation $\mathbf{V}$ ne s'annule pas forcément lorsque $\dot{\underline{\mathbf{h}}} = 0$.

En effet, ceci peut subvenir quand $\underline{\mathbf{u}}$ est constant mais non égal à $\underline{\mathbf{u}}^*$. Alors, lorsque $\underline{\mathbf{u}}$, $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ ont été amenés par la commande dans une configuration coplanaire, la différence $\underline{\mathbf{h}} - \underline{\mathbf{h}}^*$ contient une composante selon $\underline{\mathbf{h}}$. Cette composante n'est nulle que lorsque $\underline{\mathbf{h}} = \underline{\mathbf{h}}^*$ car ces deux vecteurs sont unitaires. Lorsqu'elle n'est pas nulle, $\mathbf{V}$ possède donc une composante le long de $\underline{\mathbf{u}} \times \underline{\mathbf{h}}$ qui ne varie pas. En effet, $\underline{\mathbf{u}}$ est fixe; quant à $\underline{\mathbf{h}}$, il est maintenu à l'intersection du plan fixe $(\underline{\mathbf{u}}, \underline{\mathbf{h}}^*)$ et du plan perpendiculaire à $\underline{\mathbf{u}}$.

Cela signifie que le robot continue à se déplacer après la minimisation de l'écart entre $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ et bien que l'image de la droite reste fixe. On court donc le risque de voir le robot diverger. Moins dramatiquement, cela signifie surtout que certains mouvements du robot ne peuvent pas être observés et que leur commande n'a pas grand sens.

Pour pallier ce problème, on introduit l'erreur :

$$\epsilon = (\underline{\mathbf{u}} \times \underline{\mathbf{h}})^T \underline{\mathbf{h}}^* \tag{2.21}$$

et la commande en vitesse :

$$\mathbf{V} = -\mu_h \epsilon \underline{\mathbf{h}} \tag{2.22}$$

qui s'annulent simultanément lorsque $\underline{\mathbf{u}}$, $\underline{\mathbf{h}}$ et $\underline{\mathbf{h}}^*$ sont coplanaires.

Cette condition d'annulation est donc identique à celle d'annulation de $\dot{\underline{\mathbf{h}}}$, puisque celle-ci reste inchangée par rapport à (2.10)

$$\dot{\underline{\mathbf{h}}} = \frac{\mu_h}{h} \epsilon (\underline{\mathbf{u}} \times \underline{\mathbf{h}}) \tag{2.23}$$

En reconsidérant la preuve du Lemme 4, on vérifie alors que les mêmes conclusions s'imposent :

Lemme 6 (Lemme de convergence)

Lorsque la commande en rotation définie au Lemme 1 a convergé, et dans le cas idéal où la commande en translation définie en (2.22) est parfaite (c.-à-d. n'engendre pas de rotations parasites), alors cette dernière est asymptotiquement stable si la condition initiale $\underline{\mathbf{h}}(t=0) \neq -\underline{\mathbf{h}}^*$ *est vérifiée.*
Cette condition représente un équilibre instable.

Le comportement de cette nouvelle loi de commande est donc identique dans l'image à celui de la commande CPN non projetée. En revanche, $\mathbf{V}$ est, par rapport à cette dernière, projetée sur la droite dirigée par $\underline{\mathbf{h}}$. En effet, cette projection est définie par $\underline{\mathbf{h}}^T(\underline{\mathbf{u}} \times (\underline{\mathbf{h}} - \underline{\mathbf{h}}^*))\underline{\mathbf{h}}$, expression égale à $\epsilon\underline{\mathbf{h}}$. Ceci explique donc l'épithète dont cette nouvelle commande est affublée.

L'intérêt de projeter $\mathbf{V}$ sur $\underline{\mathbf{h}}$ est qu'on supprime les composantes de $\mathbf{V}$ qui laissent $\underline{\mathbf{h}}$ inchangé. Tous les mouvements de translation sont donc désormais observables, ce qui est, ma foi, bien agréable pour un automaticien.

Plusieurs droites

Dans le cas de plusieurs droites, le Théorème 2 se réécrit pour la commande projetée, sous la même hypothèse que la commande en translation n'engendre pas de rotation résiduelle, en :

Théorème 3 (Condition de stabilité)

La commande en translation définie par :

$$\mathbf{V} = -\mu_h \mathbf{C}_h \begin{pmatrix} \epsilon_1 \underline{\mathbf{h}}_1 \\ \vdots \\ \epsilon_n \underline{\mathbf{h}}_n \end{pmatrix}, \quad \mu_h > 0$$

à $\underline{\mathbf{u}}_i$ $(i = 1..n)$ *constants mais non forcément égaux aux* $\underline{\mathbf{u}}_i^*$ *et avec*

$$\forall i = 1..n, \epsilon_i = (\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i)^T \underline{\mathbf{h}}_i^*$$

réduit (au sens large) les erreurs ϵ_i*, si le produit* $\mathbf{UC}_h$ *est symétrique semi défini positif avec* $\mathbf{U}^T = (\mathbf{I}_3 \cdots \mathbf{I}_3)$.

Asservissement visuel à partir de droites

Preuve :

On prend la fonction de Lyapunov :

$$L_h = \frac{1}{2}\sum_{i=1}^{n} \|\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*\|^2$$

dont la dérivée est

$$\dot{L}_h = -\sum_{i=1}^{n} \dot{\underline{\mathbf{h}}}_i^T \underline{\mathbf{h}}_i^*$$

En notant $\mathbf{E}_h = \begin{pmatrix} \epsilon_1 \underline{\mathbf{h}}_1 \\ \vdots \\ \epsilon_n \underline{\mathbf{h}}_n \end{pmatrix}$ et en faisant les remplacements d'usage, on aboutit à :

$$\dot{L}_h = -\mu_h \mathbf{E}_h^T \mathbf{C}_h^T \sum_{i=1}^{n} \frac{\epsilon_i \underline{\mathbf{h}}_i}{h_i}$$

qui s'écrit sous forme matricielle :

$$\dot{L}_h = -\mu_h \mathbf{E}_h^T \mathbf{C}_h^T \mathbf{U}^T \mathbf{W} \mathbf{E}_h$$

où la matrice $\mathbf{W}$ est la matrice diagonale formée par les facteurs $1/h_i > 0$ et est donc définie positive.

La condition de stabilité asymptotique ($\mathbf{C}_h^T \mathbf{U}^T \mathbf{W} > 0$) n'étant pas réalisable du fait de la structure de $\mathbf{U}$ (cf. Proposition 2), force est de nous contenter de prouver le caractère décroissant au sens large de la commande : $\mathbf{C}_h^T \mathbf{U}^T \mathbf{W}$ doit être semi définie positive. Comme $\mathbf{W}$ est définie positive, on obtient la condition de l'énoncé. □

Contrairement à la commande non projetée, la commande projetée possède donc une condition indépendante de l'état du système. Elle est ainsi inconditionnellement décroissante par un choix correct de la matrice de pondération $\mathbf{C}_h$.

En particulier, le choix $\mathbf{C}_h = \mathbf{U}^T$, qui implique

$$\mathbf{V} = -\mu_h \sum_{i=1}^{n} [(\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i)^T \underline{\mathbf{h}}_i^*] \underline{\mathbf{h}}_i,$$

garantit inconditionnellement que l'erreur est décroissante au sens large, quelle que soit la configuration des droites. En revanche, il existe alors deux types d'états stationnaires :

$$\sum_{i=1}^{n} [(\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i)^T \underline{\mathbf{h}}_i^*] \underline{\mathbf{h}}_i = 0 \quad et \quad \sum_{i=1}^{n} \frac{(\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i)^T \underline{\mathbf{h}}_i^*}{h_i} \underline{\mathbf{h}}_i = 0$$

qui dépendent de la configuration des droites.

Le premier type n'est que moyennement gênant car il correspond à l'annulation de la vitesse. Tomber dans cette singularité arrête donc le robot. Même si l'objectif n'est pas atteint, c'est une singularité sûre puisque le robot ne risque alors plus de diverger.

On peut montrer que cette singularité n'est pas un minimum local et qu'elle est donc instable. En effet, la dérivée seconde de L_h vaut, dans ce cas, $\ddot{L}_h = (\sum \frac{\epsilon_i \underline{\mathbf{h}}_i}{h_i})^T \dot{V}$. Si l'on n'est pas simultanément dans le second type d'état stationnaire, alors il existe un chemin pour sortir de cette position.

Le second type est plus embêtant. En effet, dans une telle singularité, la vitesse du robot ne s'annule pas forcément. Il risque donc de se déplacer de manière incontrôlée. Ce cas reste à étudier.

Enfin, on remarquera que le choix $\mathbf{C}_h = \mathbf{H}^T$ dans la commande non projetée génère la même commande effective que le choix $\mathbf{C}_h = \mathbf{U}^T$ de la commande projetée. Ce résultat est consistant avec le Théorème 2 et la Proposition 3, ce qui est naturel puisque, à bien y regarder l'opérateur $\mathbf{H}$ contient les opérateurs de projection utilisés dans la définition de la commande projetée.

2.4 Conclusion

Nous avons défini deux commandes basées sur les coordonnées de Plücker binormées : commande CPN et commande CPN projetée. Elles permettent toutes deux un découplage partiel entre translation et rotation. Ce découplage a été obtenu par le biais de l'étude de deux commandes élémentaires : une commande en rotation seule et une commande purement translationnelle. La première permet d'aligner la direction des droites sur celle qu'on souhaite qu'elles aient. La seconde permet de minimiser, à direction constante, une erreur entre les droites image courantes et désirées.

Dans le cas d'une droite, nous avons montré la convergence de chacune de ces deux commandes élémentaires et plusieurs manières de les combiner pour obtenir une convergence globale du système. Dans ce cas, la commande CPN et la commande CPN projetée se différencient par l'annulation ou non de la vitesse de translation à l'obtention de l'erreur minimale dans l'image.

Dans le cas de plusieurs droites, commande CPN projetée ou non partagent la même conception en 2 phases : alignement des directions des droites sur les directions désirées par une commande en rotation seulement; puis alignement des droites dans l'image par une commande en translation tout en maintenant les directions désirées par la commande en rotation (idéalement nulle) de la première phase.

Dans les deux commandes, la première phase est identique. De plus, nous avons pu étendre les résultats de convergence pour une commande en rotation consistant

tout simplement à additionner les commandes élémentaires construites pour chaque droite considérée individuellement.

Pour ce qui est de la deuxième phase, toujours dans le cas de plusieurs droites, seuls des résultats de convergence partielle ont été exhibés. En effet, nous avons prouvé que la commande CPN projetée est inconditionnellement bornée lorsqu'on la construit identiquement par sommation des influences de chaque droite. Nous avons aussi identifié les états stationnaires de cette commande, qui restent cependant à étudier plus en détail. Enfin, pour la commande non projetée, nous n'avons pu qu'obtenir une condition dépendant de l'état dans lequel se trouve le système.

Nous allons toutefois voir dans le chapitre suivant que l'on peut prouver la stabilité asymptotique de ces deux commandes dans le cas particulier d'un trièdre orthogonal.

Chapitre 3

Positionnement par rapport à un trièdre orthogonal

3.1 Introduction

Dans ce chapitre, nous allons nous consacrer à l'étude d'un cas particulier des commandes présentées au chapitre précédent. Il s'agira de positionner une caméra par rapport à un objet modélisé par un trièdre orthogonal. L'intérêt d'une telle configuration est qu'elle se rencontre souvent dans les environnements structurés créés par l'homme, cibles privilégiées de la robotique.

Un premier exemple d'une telle configuration concerne les robots mobiles. Ces robots n'ont connaissance de leur position que par odométrie, c.-à-d. par intégration de leurs déplacements. Les déplacements réels du robot ne correspondant pas exactement avec ses déplacements mesurés, des erreurs s'accumulent au cours du mouvement. Les robots mobiles perdent ainsi progressivement le sens de l'orientation. Pour compenser cette dérive, ils doivent procéder régulièrement à un recalage : ils retournent se placer dans une position précisément connue. En particulier, les coins des pièces d'un bâtiment peuvent, de manière générique, fournir de telles positions.

Un deuxième exemple est lié aux tâches d'inspection. Dans de telles applications, un robot doit effectuer un parcours le long d'un objet, le plus souvent manufacturé. Il doit donc initialement se placer en un point de départ qui n'est, en général, pas connu avec assez de précision pour utiliser une commande en boucle ouverte. Le positionnement initial doit alors se faire par une boucle fermée sur des données

capteurs. De nombreux objets possédant des coins droits, on peut utiliser ces derniers comme point de départ.

Une autre application possible du positionnement par rapport à un trièdre consiste en le transbordement de containers d'un cargo vers un train. En effet, ces containers sont parallélépipèdes rectangles et possèdent donc des coins par rapport auxquels on peut positionner un robot.

Enfin, le dernier exemple est le soudage de pièces de bateau. Dans cette application, qui sera détaillée plus loin (§3.4) et nous servira de base expérimentale, il faut souder, deux à deux, trois plaques mutuellement orthogonales, donc formant un trièdre orthogonal.

Auparavant, nous donnerons quelques résultats théoriques supplémentaires, directement liés à la configuration particulière que forme la jonction de trois droites orthogonales. Nous verrons ainsi qu'elle forme un cas dégénéré pour les algorithmes de vision. Parmi les solutions pratiques qui permettent de le compenser, nous choisirons la plus réutilisable de toutes : l'adjonction d'un pointeur laser à la caméra. Ce nouveau « mécanisme de vision[1] » sera étudié puis utilisé dans la commande.

3.2 Résultats théoriques

Dhome *et al.* [DRLR89] ont montré qu'une jonction de droites est un cas dégénéré pour le calcul de pose. Il est en effet impossible d'estimer la profondeur de l'intersection commune de ces droites (que nous appellerons *centre du faisceau* par la suite) à partir d'une seule image. Du point de vue de la commande, cela signifie que la distance du robot à la pièce n'est pas observable avec une seule caméra. Il est donc impossible avec la seule image courante de contrôler cette distance.

Nous allons donc chercher une solution pour observer cette distance. Cette solution doit être la plus générale possible. Ainsi, nous exclurons les solutions dédiées à notre application de soudage de pièces de bateau telles que : utilisation de points de pré-soudure entre les plaques, de la texture de ces dernières ou encore mesure de l'interstice entre les plaques.

3.2.1 Stéréovision et faisceau de droites

Écartons immédiatement l'idée de faire une reconstruction euclidienne comme on pourrait le faire pour des points, car, dans le cas des droites, ce n'est pas moins de 6 droites qu'il faut observer dans 3 images (voir l'état de l'art).

Pour observer la distance, on peut alors envisager d'utiliser un mélange de calcul de pose et de reconstruction, afin de retrouver cette distance, ou, du moins, une distance relative entre deux (ou plus) vues.

1. Merci à Andreas Ruf pour avoir inventé ce terme.

Donnons-nous donc deux images d'un même trièdre et calculons la pose (au facteur d'échelle près) associée à chacune de ces deux images. Nous obtenons alors $(\mathbf{R}, \lambda\mathbf{t})$ et $(\mathbf{R}', \lambda'\mathbf{t}')$ où λ et λ' sont les distances inconnues (avec la convention que $\mathbf{t}$ et $\mathbf{t}'$ sont ici unitaires). On peut ainsi exprimer le déplacement entre ces deux vues $(\mathbf{R}_d, \mathbf{t}_d)$ en fonction de λ et λ' : $(\mathbf{R}_d, \mathbf{t}_d) = (\mathbf{R}'\mathbf{R}^T, -\lambda\mathbf{R}_d\mathbf{t} + \lambda'\mathbf{t}')$.

Mouvement euclidien connu

Si l'on dispose du mouvement euclidien de la caméra $\mathbf{R}_A, \mathbf{t}_A$, le tour est joué car alors, nous n'avons plus qu'à écrire $\mathbf{R}_A = \mathbf{R}_d$ et $\mathbf{t}_A = \mathbf{t}_d$ pour obtenir trois équations linéaires en λ et λ'. Le seul moyen pour obtenir le mouvement euclidien de la caméra est d'utiliser le mouvement du robot et l'étalonnage pince-caméra. Cependant, on ne peut jamais être certain que le mouvement mesuré du robot est son mouvement effectif. Il est, par conséquent, peu envisageable d'utiliser cette solution et nous en sommes réduits à utiliser les informations purement visuelles.

Contrainte épipolaire

On peut alors envisager d'utiliser la contrainte épipolaire. En effet, si l'on dispose des projections ($\boldsymbol{m}$ et $\boldsymbol{m}'$) d'un même point physique dans les deux images, on peut exprimer la contrainte épipolaire à partir des paramètres de formation de l'image (représentés par la matrice $\mathbf{K}$) et du déplacement euclidien entre les deux images :

$$\boldsymbol{m}'^T \mathbf{K}^{-T} As(\mathbf{t}_d)\mathbf{R}_d\mathbf{K}^{-1}\boldsymbol{m} = 0$$

qui se réécrit :

$$-\lambda \boldsymbol{m}'^T \mathbf{K}^{-T} As(\mathbf{R}_d\mathbf{t})\mathbf{R}_d\mathbf{K}^{-1}\boldsymbol{m} + \lambda' \boldsymbol{m}'^T \mathbf{K}^{-T} As(\mathbf{t}')\mathbf{R}_d\mathbf{K}^{-1}\boldsymbol{m} = 0 \qquad (3.1)$$

Nous avons donc d'une équation à deux inconnues (λ et λ'), ce qui, en général, permet d'obtenir une distance relative.

Contrainte épipolaire et centre du faisceau — Le seul point physique dont nous disposons est le centre du faisceau, dont les projections sont respectivement $\mathbf{Kt}$ et $\mathbf{Kt}'$. Lorsque l'on applique la contrainte épipolaire, on obtient donc :

$$\mathbf{t}'^T As(\mathbf{t}_d)\mathbf{R}_d\mathbf{t} = 0$$

Cette contrainte devient, en faisant apparaître les inconnues et après quelques manipulations algébriques évidentes :

$$-\lambda \mathbf{t}'^T \underbrace{As(\mathbf{R}_d\mathbf{t})\mathbf{R}_d\mathbf{t}}_{0} + \lambda' \mathbf{t}'^T As(\mathbf{t}')\mathbf{R}_d\mathbf{t} = 0$$

Or, $\mathbf{t}'^T As(\mathbf{t}')\mathbf{R}_d\mathbf{t} = \mathbf{t}'^T(\mathbf{t}' \times \mathbf{R}_d\mathbf{t}) = 0$, ce qui implique que la contrainte épipolaire est trivialement vérifiée pour le centre du faisceau. On ne peut donc pas déterminer les facteurs d'échelle inconnus à l'aide de ce point.

Asservissement visuel à partir de droites

Contrainte épipolaire et droites — Une autre idée est proche de ce qu'on trouve chez Zhang [Zha95]. Celui-ci, rappelons-le, propose de calculer la forme et le mouvement de la scène à partir de mises en correspondance de segments. Pour cela, il utilise un critère de recouvrement maximal entre segments correspondants.

Il s'agit de prendre un point particulier $\boldsymbol{m}$ sur une droite (l), projection dans la première image d'une droite 3D (L), et d'essayer de retrouver son correspondant $\boldsymbol{m'}$, sur la droite (l'), projection de (L) dans la deuxième image. On espère ainsi pouvoir trouver les points correspondants et les facteurs d'échelle.

Connaissant le modèle du trièdre, les paramètres intrinsèques de la caméra et le point $\boldsymbol{m}$ et en fixant, sans perte de généralité, λ à 1, on peut reconstruire le point 3D $\boldsymbol{M}$ situé sur (L) et se projetant en $\boldsymbol{m}$.

Pour trouver le point $\boldsymbol{m'}$, il faut placer l'image à la distance λ' sur l'axe défini par $\mathbf{t}'$. Le point $\boldsymbol{m'}$ sera alors l'intersection de la droite épipolaire (paramétrée par λ') associée à m dans la deuxième image et de la droite (l'). Malheureusement, comme illustré en Figure 3.1, il n'est pas possible de discerner λ' d'une autre valeur. En effet, pour le faire, il faut pouvoir dire où se trouve $\boldsymbol{m'}$ le long de (l'), ce que l'on cherche à savoir !

Une autre façon de voir le problème est de considérer le mouvement particulier de translation le long de la droite de vue passant par le centre du faisceau. Alors, l'épipole est situé sur cette droite de vue et la droite épipolaire est la droite (l'), elle-même. Il y a donc une infinité d'intersections possibles de ces deux droites. Il est donc impossible de définir l'amplitude de la translation effectuée.

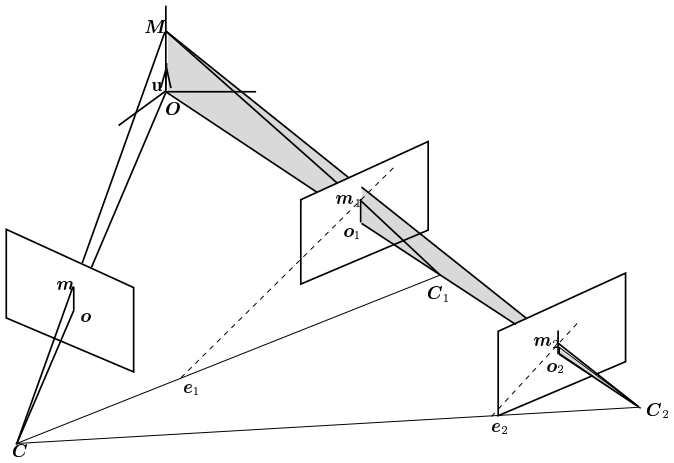


FIG. 3.1: *Comment choisir lequel de $\boldsymbol{C}_1$ ou de $\boldsymbol{C}_2$ est la position du centre optique de la caméra ?*

Remarque 1 : On peut vérifier que ce résultat est valable pour plusieurs images.

Remarque 2 : Ce résultat montre aussi qu'on ne peut pas utiliser la méthode de Zhang. Cette méthode ne peut, dans ce cas présent, que donner une solution particulière parmi l'infinité des possibles. De plus, en pratique, les segments que nous verrons dans l'image atteindront tous le bord de l'image. Et ainsi toute translation supplémentaire en direction du centre du faisceau le long de sa droite de vue laissera l'image des segments inchangée. Le recouvrement maximal des segments sera donc trivialement vérifié et ne sera alors plus discriminant.

3.2.2 Utilisation d'un laser

On ne peut observer la profondeur avec une seule caméra et la stéréovision ne peut répondre à notre besoin que si l'on dispose d'un point autre que le centre du faisceau. Or, il n'en existe pas que l'on soit sûr de retrouver dans toute application mettant en œuvre un faisceau de droites. Qu'à cela ne tienne, ajoutons-en un ! Et pour cela, utilisons un laser.

Laser fixe ou embarqué ?

Laser fixe — Cette solution est pratique car on place un pointeur laser (aussi simple que ceux utilisés dans les conférences) sur la pièce en un point quelconque (en pratique, pas trop loin du centre du faisceau). Elle est générique. Elle est simple car le *point laser*, intersection du rayon laser et de la pièce, est alors un point fixe, dont la projection dans l'image (la *tache laser*) est facilement détectable.

Pour éviter les contraintes d'étalonnage, on ne peut utiliser directement le point laser dans la commande. Cependant, comme il est fixe, on peut utiliser la contrainte épipolaire reliant les deux taches laser extraites des images pour observer une distance relative entre deux vues successives. Ainsi, si l'on connaît la distance initiale et la distance de consigne, on peut alors reconstruire la distance et l'utiliser lors de la commande.

Toutefois, il se peut qu'au cours de l'asservissement, le robot occulte le rayon laser ou que l'opération de traitement d'image n'arrive pas à détecter la tache laser. Cela impliquerait alors la perte de l'information de distance, donc l'impossibilité de savoir s'il faut encore avancer ou non. Pour pallier ce problème, deux solutions sont envisageables. La première consiste à prédire la position du point dans l'image à l'aide du torseur de commande et la seconde à conserver l'historique des vitesses (*a priori* (exponentiellement) décroissantes) de façon à interpoler la vitesse manquante. Ces deux solutions supposent que la tache laser réapparaisse assez vite et à peu près à l'endroit prévu. Comme ce n'est pas garanti, embarquons le laser sur le robot pour éviter toute occlusion de son rayon.

Laser embarqué — On peut envisager de monter un laser sur le robot de manière à ce qu'il touche la pièce en un point visible dans l'image. Désormais, le point laser se déplace sur la pièce lorsque le robot bouge. On ne peut donc pas lui appliquer la contrainte épipolaire. En revanche, on peut énoncer la proposition suivante.

Proposition 4

Lorsqu'un pointeur laser est rigidement lié à une caméra en mouvement, la tache laser se déplace sur une droite constante, qu'on appellera droite épipolaire laser.

Preuve :

On peut prouver ceci de deux manières : en considérant le faisceau laser comme l'axe optique d'une caméra virtuelle et en utilisant la géométrie épipolaire ou en considérant que le plan bouge devant le système caméra-laser (voir Figures 3.2 et 3.3). □

Par la suite, nous ne traiterons que le cas d'un laser embarqué qui offre de nouveaux résultats théoriques.

Étalonnages

Si l'on sait où se trouve le rayon laser par rapport à la caméra, on peut de plus obtenir la profondeur du point laser par triangulation et par suite, la distance du centre du faisceau de droites (voir Annexe B).

Cependant, nous verrons lors de la commande (§3.2.3, p.83) qu'un tel *étalonnage laser-caméra* est inutile et que l'on peut se contenter d'un *étalonnage image*, consistant à trouver la position de la droite épipolaire laser. Toutefois, nous supposerons jusque là l'étalonnage laser-caméra acquis.

Il n'est pas nécessaire de dédier une annexe à la description de l'étalonnage image, puisque celle-ci tient en quelques lignes. En effet, les mouvements de la caméra font se déplacer la tache laser le long de la droite épipolaire laser. Pour obtenir cette dernière, il suffit donc de déplacer la caméra, aléatoirement ou non, d'enregistrer quelques positions de la tache laser et de faire une simple régression linéaire.

Abscisse curviligne

Notons $\boldsymbol{P} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$, les coordonnées du point laser dans le repère caméra. Notons $\boldsymbol{p} = \begin{pmatrix} u \\ v \end{pmatrix}$, la tache laser dans l'image. Notons (d_L) la droite épipolaire laser d'équation : $ax + by + c = 0$ avec $a^2 + b^2 = 1$ [2], et $\mathbf{v} = \begin{pmatrix} -b \\ a \end{pmatrix}$, son vecteur directeur.

2. Comme cette droite ne sera pas utilisée en 3D, il n'est pas nécessaire de prendre la convention $a^2 + b^2 + c^2 = 1$ des coordonnées de Plücker.

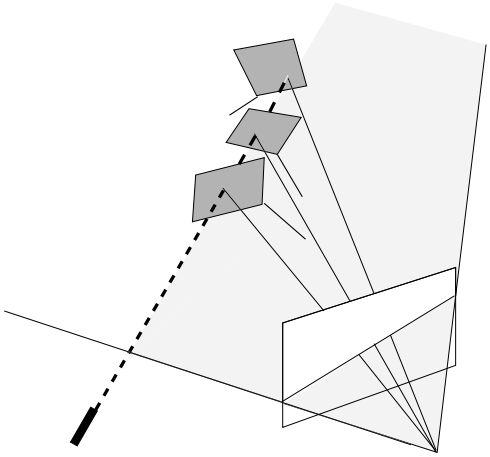


FIG. 3.2: *La projection dans l'image d'un rayon laser fixe par rapport à la caméra est une droite constante.*

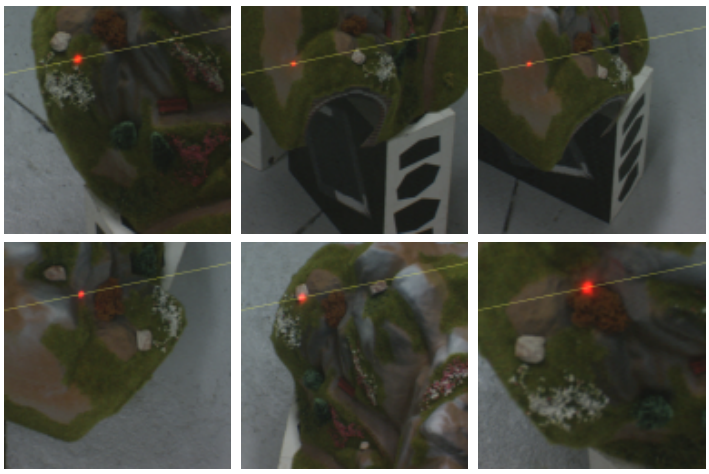


FIG. 3.3: *La preuve par l'image*

Définissons enfin des notations pour les paramètres de l'étalonnage laser-caméra exprimés dans le repère de la caméra (voir en Annexe B pour plus de détails). L'origine du rayon laser, placée à l'intersection du plan $z = 0$ du repère caméra, est notée $\mathbf{t}_L = h_L(-b, a, 0)^T$. Le vecteur directeur du rayon est noté $\mathbf{k}_L$. Enfin, on construit un repère canonique du faisceau laser en prenant pour base $(\mathbf{i}_L, \mathbf{j}_L, \mathbf{k}_L)$ où $\mathbf{i}_L = (a, b, c)^T$ et $\mathbf{j}_L = \mathbf{k}_L \times \mathbf{i}_L$. Notons finalement z_L la distance du point laser $\boldsymbol{P}$ à l'origine du rayon laser.

Comme la tache laser se déplace le long de la droite $(\mathtt{d}_L)$, on peut envisager de caractériser la tache laser $\boldsymbol{p}$ par une abscisse curviligne. Pour cela, exprimons le fait que le point laser $\boldsymbol{P}$ appartient au faisceau laser par :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{t}_L + z_L \mathbf{k}_L$$

La projection $\boldsymbol{p}$ de ce point $\boldsymbol{P}$ dans l'image est :

$$\begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} = \frac{\mathbf{t}_L}{z} + \frac{z_L}{z}\mathbf{k}_L = \frac{h_L}{z}\begin{pmatrix} -b \\ a \\ 0 \end{pmatrix} + \frac{z_L}{z}\begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix}$$

La troisième ligne de ces équations nous donne : $\frac{z_L}{z}k_z = 1$, ce qui nous permet d'obtenir en définitive :

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{z_L}\frac{h_L}{k_z}\begin{pmatrix} -b \\ a \end{pmatrix} + \frac{1}{k_z}\begin{pmatrix} k_x \\ k_y \end{pmatrix}$$

En posant $\begin{pmatrix} u_0 \\ v_0 \end{pmatrix} = \frac{1}{k_z}\begin{pmatrix} k_x \\ k_y \end{pmatrix}$ et $s = \frac{1}{z_L}\frac{h_L}{k_z}$, nous pouvons exprimer la position du point $\boldsymbol{p}$ à l'aide de l'abscisse curviligne s :

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + s \begin{pmatrix} -b \\ a \end{pmatrix} \tag{3.2}$$

où le point $(u_0, v_0)^T$ est un point quelconque de la droite.

Équation du mouvement

Le calcul de la vitesse revient donc à calculer la dérivée temporelle de l'abscisse curviligne s puisque d'après (3.2), $\frac{d}{dt}\begin{pmatrix} u \\ v \end{pmatrix} = \frac{ds}{dt}\,\mathbf{v}$.

Le calcul de $\frac{ds}{dt}$ se ramène à celui de $\frac{d\ z_L}{dt}$ par :

$$\frac{ds}{dt} = \frac{d}{dt}\left(\frac{1}{z_L}\frac{h_L}{k_z}\right) = -\frac{h_L}{z_L^2 k_z}\frac{d\ z_L}{dt}$$

Notons à présent $\bar{r}$, la position/orientation du repère laser exprimé dans le repère caméra et $\bar{r}_L$, la même attitude, exprimée cette fois-ci dans le repère laser. Alors :

$$\frac{dz_L}{dt} = \frac{\partial\, z_L}{\partial \bar{r}}\tau = \frac{\partial\, z_L}{\partial \bar{r}_L}\frac{\partial \bar{r}_L}{\partial \bar{r}}\tau$$

où τ est le torseur cinématique de la caméra et $\frac{\partial \bar{r}_L}{\partial \bar{r}} = \Theta_L = \begin{pmatrix} \mathbf{R}_L & As(\mathbf{t}_L) \\ \mathbf{0} & \mathbf{R}_L \end{pmatrix}$ avec $\mathbf{R}_L = (\,\mathbf{i}_L\ \mathbf{j}_L\ \mathbf{k}_L\,)$.

Scène plane — Considérons le cas où le laser se déplace tout en intersectant un même plan de vecteur normal $\mathbf{n}$. Dans ce cas, la matrice $\frac{\partial\, z_L}{\partial \bar{r}_L}$, qui représente la variation de la distance du point laser à l'origine du faisceau laser en fonction des mouvement de ce dernier, est égale à :

$$\frac{\partial\, z_L}{\partial \bar{r}_L} = \begin{pmatrix} \tan\alpha_x & \tan\alpha_y & 1 & -z_L\tan\alpha_y & z_L\tan\alpha_x & 0 \end{pmatrix} \tag{3.3}$$

où α_x est l'angle entre la projection $\mathbf{n}_x$ du vecteur $\mathbf{n}$ sur le plan $(\mathbf{i}_L, \mathbf{k}_L)$ et le vecteur $\mathbf{k}_L$ et α_y est l'angle entre $\mathbf{n}_y$, projection de $\mathbf{n}$ sur $(\mathbf{j}_L, \mathbf{k}_L)$, et $\mathbf{k}_L$.

Avant d'interpréter ce résultat, notons qu'il est identique à celui fourni par la formule (7.1.48) de [SLBE91, pp 273–274] :

$$\frac{\partial\, z_L}{\partial \bar{r}_L} = -\frac{1}{\mathbf{n}^T\mathbf{k}_L}\left(\,\mathbf{n}^T \mid (\mathbf{k}_L\times\mathbf{n})^T\,\right).$$

Pour cela, il suffit d'écrire les coordonnées de $\mathbf{n}$, paramétrées par les angles α_x et α_y :

$$\mathbf{n} = \begin{pmatrix} \sin\alpha_x\cos\alpha_y w \\ \cos\alpha_x\sin\alpha_y w \\ \cos\alpha_x\cos\alpha_y w \end{pmatrix},$$

où $w = (\,\sqrt{\cos^2\alpha_y + \sin^2\alpha_y\cos^2\alpha_x}\,)^{-1}$ est un facteur de normalisation.

Le troisième et sixième coefficients de (3.3) sont triviaux puisqu'ils correspondent respectivement aux mouvements de translation le long du faisceau et de rotation autour de ce faisceau. Pour calculer les coefficients restants, aidons-nous de la Figure 3.4. Grâce à elle, nous pouvons calculer $\frac{\partial z_L}{\partial x}$, la dérivée partielle de z_L pour un mouvement de translation le long de l'axe X du repère laser (c.-à-d. l'axe défini par l'origine $\mathbf{t}_L$ et le vecteur $\mathbf{i}_L$ choisi) et $\frac{\partial z_L}{\partial \theta_x}$, la dérivée partielle de z_L pour un mouvement de rotation autour de l'axe X.

La partie gauche de la Figure 3.4 nous permet d'écrire :

$$\frac{\partial z_L}{\partial x} = \tan\alpha_y \tag{3.4}$$

Asservissement visuel à partir de droites

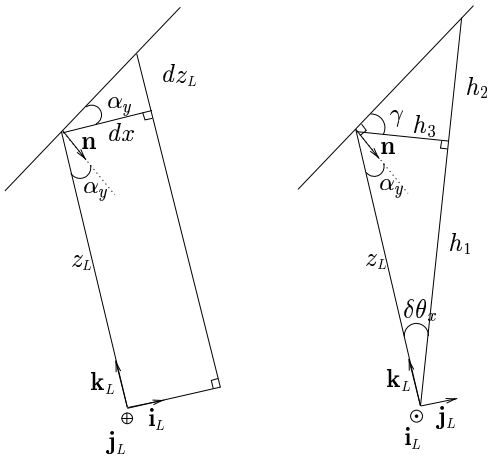


FIG. 3.4: *Variation du faisceau laser lors d'une translation selon l'axe X (gauche) et d'une rotation autour de cet axe (droite)*

Sa partie droite nous donne :

$$\begin{aligned} h_1 &= z_L \cos\delta\theta_x \\ h_3 &= z_L \sin\delta\theta_x \\ h_2 &= h_3 \tan\gamma \\ \gamma &= \alpha_y + \delta\theta_x \\ dz_L &= h_1 + h_2 - z_L \end{aligned}$$

d'où $dz_L = z_L(\cos\delta\theta_x + \sin\delta\theta_x \tan(\alpha_y + \delta\theta_x) - 1)$, qui en faisant tendre $\delta\theta_x$ vers 0[3], nous donne :

$$\frac{\partial z_L}{\partial \theta_x} = z_L \tan\alpha_y \tag{3.5}$$

Ces résultats se généralisent à $\frac{\partial z_L}{\partial y}$ et $\frac{\partial z_L}{\partial \theta_y}$. On remarquera l'apparition d'un signe '$-$' dans l'expression de cette dernière par rapport à (3.5). Ce changement est dû à une orientation différente des angles dans les plans $(\mathbf{i}_L, \mathbf{k}_L)$ et $(\mathbf{j}_L, \mathbf{k}_L)$.

La matrice d'interaction reliant la vitesse curviligne et le torseur cinématique, toutes quantités exprimées dans le repère caméra, est :

$$\frac{ds}{dt} = -\frac{h_L}{k_z z_L^2}\begin{pmatrix}\tan\alpha_x & \tan\alpha_y & 1 & -z_L\tan\alpha_y & z_L\tan\alpha_x & 0\end{pmatrix}\Theta_L\tau \tag{3.6}$$

ou bien, en faisant intervenir la profondeur z du point laser le long de sa droite de vue :

$$\frac{ds}{dt} = -\frac{h_L k_z}{z^2}\begin{pmatrix}\tan\alpha_x & \tan\alpha_y & 1 & -\frac{z}{k_z}\tan\alpha_y & \frac{z}{k_z}\tan\alpha_x & 0\end{pmatrix}\Theta_L\tau \tag{3.7}$$

3. On vérifie que cette limite est valide même lorsque $\alpha_y = 0$,

Remarque : Si la vitesse de rotation $\mathbf{\Omega}$ est nulle (c.-à-d. que l'on peut assurer l'orientation du trièdre) et si la vitesse de translation $\mathbf{V}$ est de direction constante $\underline{\mathbf{u}}_V$, alors la vitesse dans l'image est proportionnelle à : $\frac{v}{z^2}$ où v est tel que $\mathbf{V} = v\underline{\mathbf{u}}_V$ et le coefficient de proportionnalité est de signe constant. En effet, l'équation (3.7) se réduit à :

$$\frac{ds}{dt} = -\frac{v}{z^2} \underbrace{\left[h_L k_z \begin{pmatrix} \tan\alpha_x & \tan\alpha_y & 1 \end{pmatrix} \mathbf{R}_L \underline{\mathbf{u}}_V\right]}_{=cte} \tag{3.8}$$

3.2.3 Commande

La commande que nous proposons se décompose en deux commandes élémentaires. La première, basée sur la seule observation des droites de l'image, permet l'obtention de la pose partielle désirée. La seconde, basée sur l'observation de la tache laser, permet quant à elle l'obtention de la profondeur désirée.

Pose partielle

Ce sont en fait deux commandes élémentaires que nous proposons, issues respectivement des commandes CPN projetée et non projetée.

Commande non projetée — Rappelons les lois de la commande CPN non projetée, appliquées au cas du trièdre orthogonal. En particulier, on utilise le résultat du Théorème 1 pour définir la commande en rotation :

$$\begin{aligned} \mathbf{\Omega} &= \mu_u \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*, \quad \mu_u > 0 \\ \mathbf{e}_{h_i} &= \underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*) \\ \mathbf{V} &= -\mu_h \mathbf{C}_h \begin{pmatrix} \mathbf{e}_{h_1} \\ \vdots \\ \mathbf{e}_{h_n} \end{pmatrix}, \quad \begin{cases} \mu_h = 0 & si \ \|\mathbf{\Omega}\| \leq \epsilon \\ \mu_h > 0 & sinon \end{cases} \end{aligned}$$

Il nous faut donc choisir une matrice de combinaison $\mathbf{C}_h$. En particulier, nous allons nous intéresser au choix $\mathbf{C}_h = \mathbf{U}^T$, où $\mathbf{U}$, définie au chapitre précédent, correspond à sommer chacune des composantes $\mathbf{e}_{h_i}$. La commande en translation devient alors :

$$\mathbf{V} = -\mu_h \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*) \tag{3.9}$$

Ce choix ne satisfait pas la condition suffisante de stabilité générale (Théorème 2), et pourtant, il garantit la stabilité asymptotique du système vers la pose partielle désirée, après convergence en rotation.

Asservissement visuel à partir de droites

Pour voir cela, étudions l'effet de cette commande, après cette convergence. L'équation du mouvement de la droite j dans l'image (1.18) s'écrit alors :

$$\dot{\underline{\mathbf{h}}}_j = -\mu_h \frac{\mathbf{P}_j}{h_j} \left((\sum_{i=1}^{3} \underline{\mathbf{u}}_i \times \mathbf{e}_j) \times \underline{\mathbf{u}}_j \right), \quad j = 1..3 \tag{3.10}$$

où l'on a noté $\mathbf{P}_j$ l'opérateur de projection $\mathbf{I}_3 - \underline{\mathbf{h}}_j \underline{\mathbf{h}}_j^T$ et $\mathbf{e}_j = \underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*$.

Après convergence, on a

$$\underline{\mathbf{u}}_j = \underline{\mathbf{u}}_j^*, \quad j = 1..3$$

dont on déduit :

$$\underline{\mathbf{u}}_j^T \mathbf{e}_j = 0, \quad j = 1..3 \tag{3.11}$$

En utilisant ce résultat et l'orthogonalité deux à deux des droites, on obtient :

$$(\underline{\mathbf{u}}_i \times \mathbf{e}_j) \times \underline{\mathbf{u}}_j = \begin{cases} \mathbf{e}_j & i = j \\ -(\underline{\mathbf{u}}_j^T \mathbf{e}_i)\underline{\mathbf{u}}_i & i \neq j \end{cases} \tag{3.12}$$

Soit à présent la fonction de Lyapunov

$$L = \frac{1}{2} \sum_{j=1}^{3} \mathbf{e}_j^T \mathbf{e}_j$$

Alors, sa dérivée $\dot{L}$ s'écrit avec (3.10) et (3.12) :

$$\dot{L} = -\mu_h \sum_{j=1}^{3} \frac{1}{h_j} (\mathbf{e}_j^T \mathbf{P}_j \mathbf{e}_j - \mathbf{e}_j^T \mathbf{P}_j \sum_{i \neq j} (\underline{\mathbf{u}}_j^T \mathbf{e}_i) \underline{\mathbf{u}}_i) \tag{3.13}$$

On définit :

$$E = \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{pmatrix}, \quad U = \begin{pmatrix} \underline{\mathbf{u}}_1 \\ \underline{\mathbf{u}}_2 \\ \underline{\mathbf{u}}_3 \end{pmatrix}$$

et l'opérateur :

$$\mathbf{P}_U = \frac{1}{2}(\mathbf{I}_9 - U \otimes U^T) = \begin{pmatrix} \mathbf{I}_3 - \underline{\mathbf{u}}_1 \underline{\mathbf{u}}_1^T & -\underline{\mathbf{u}}_2 \underline{\mathbf{u}}_1^T & -\underline{\mathbf{u}}_3 \underline{\mathbf{u}}_1^T \\ -\underline{\mathbf{u}}_1 \underline{\mathbf{u}}_2^T & \mathbf{I}_3 - \underline{\mathbf{u}}_2 \underline{\mathbf{u}}_2^T & -\underline{\mathbf{u}}_3 \underline{\mathbf{u}}_2^T \\ -\underline{\mathbf{u}}_1 \underline{\mathbf{u}}_3^T & -\underline{\mathbf{u}}_2 \underline{\mathbf{u}}_3^T & \mathbf{I}_3 - \underline{\mathbf{u}}_3 \underline{\mathbf{u}}_3^T \end{pmatrix}$$

On montre aisément que $\mathbf{P}_U$ est un projecteur orthogonal sur le noyau de U^T, donc sur l'hyperplan Π orthogonal à U. En écrivant dans (3.13)

$$(\underline{\mathbf{u}}_j^T \mathbf{e}_i)\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_i (\underline{\mathbf{u}}_j^T \mathbf{e}_i) = (\underline{\mathbf{u}}_i \underline{\mathbf{u}}_j^T)\mathbf{e}_i$$

on obtient

$$\dot{L} = -2\mu_h \tilde{E}^T \mathbf{P}_U E \tag{3.14}$$

où $\tilde{E}$ est le vecteur formé de la concaténation des $\tilde{\mathbf{e}}_j = \frac{1}{h_j}\mathbf{P}_j\mathbf{e}_j$ pour $j = 1..3$.

D'après (3.11), on trouve que E et U sont orthogonaux, d'où $E \in \Pi$ et $\mathbf{P}_U E = E$. Cela simplifie l'expression de $\dot{L}$ en :

$$\dot{L} = -2\mu_h \tilde{E}^T E = -2\mu_h \sum_{j=1}^{3} \tilde{\mathbf{e}}_j^T \mathbf{e}_j$$

En utilisant l'idempotence des projections $\mathbf{P}_j$, la somme précédente prend la forme :

$$\dot{L} = -2\mu_h \sum_{j=1}^{3} \frac{1}{h_j} \mathbf{e}'^T_j \mathbf{e}'_j$$

où $\mathbf{e}'_j = \mathbf{P}_j\mathbf{e}_j$. Par conséquent, cette somme est négative et ne s'annule que lorsque tous les $\mathbf{e}'_j$ s'annulent, c.-à-d. si les $\mathbf{e}_j$ sont nuls ou colinéaires aux $\underline{\mathbf{h}}_j$. Ce dernier cas n'est possible, par définition des $\mathbf{e}_j$, que pour $\underline{\mathbf{h}}_j = \pm\underline{\mathbf{h}}^*_j$.

Ceci se résume par :

Théorème 4 (Théorème de convergence)

La loi de commande en deux phases :

$$\begin{aligned} \mathbf{\Omega} &= \mu_u \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}^*_i, \quad \mu_u > 0 \\ \mathbf{V} &= -\mu_h \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}^*_i), \quad \begin{cases} \mu_h = 0 & si\ \|\mathbf{\Omega}\| \leq \epsilon \\ \mu_h > 0 & sinon \end{cases} \end{aligned}$$

est asymptotiquement stable en dehors des singularités initiales suivantes :

- *pour la première phase (rotation) :* $\exists j, \underline{\mathbf{u}}_j = -\underline{\mathbf{u}}^*_j$;
- *pour la deuxième phase (translation) :* $\exists j, \underline{\mathbf{h}}_j = -\underline{\mathbf{h}}^*_j$.

Commande projetée — Les théorèmes généraux du chapitre précédent garantissent la décroissance de cette commande, mais exhibent des singularités. Le théorème suivant permet de caractériser ces singularités, et par là prouve la convergence

de la commande, dans le cas du trièdre orthogonal.

Théorème 5 (Théorème de convergence)

La commande CPN projetée en deux phases, définie par :

$$\begin{aligned} \mathbf{\Omega} &= \mu_u \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*, \quad \mu_u > 0 \\ \mathbf{V} &= -\mu_h \sum_{i=1}^{3} \left[(\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i)^T \underline{\mathbf{h}}_i^* \right] \underline{\mathbf{h}}_i, \quad \begin{cases} \mu_h = 0 & si \ \|\mathbf{\Omega}\| \leq \epsilon \\ \mu_h > 0 & sinon \end{cases} \end{aligned}$$

est asymptotiquement stable si la position désirée de la caméra est dans le même octant que la position initiale car les singularités définies par : $\sum_{i=1}^{3} \epsilon_i \underline{\mathbf{h}}_i = 0$ *et* $\sum_{i=1}^{3} \frac{\epsilon_i \underline{\mathbf{h}}_i}{h_i} = 0$ *correspondent aux cas dégénérés où la caméra voit le trièdre par la tranche.*

Preuve :

1. Prouvons que

$$\sum_{i=1}^{3} \epsilon_i \underline{\mathbf{h}}_i = 0 \tag{3.15}$$

implique que tous les $\epsilon_i = 0$.

Le système (3.15) se développe en :

$$\begin{pmatrix} 0 & \underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_1 & \underline{\mathbf{h}}_3^T \underline{\mathbf{u}}_1 \\ \underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_2 & 0 & \underline{\mathbf{h}}_3^T \underline{\mathbf{u}}_2 \\ \underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_3 & \underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_3 & 0 \end{pmatrix} \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix} = 0 \tag{3.16}$$

Remarquons d'abord que s'il existe deux indices $i \neq j$ tels que $\underline{\mathbf{h}}_i^T \underline{\mathbf{u}}_j = 0$, alors les droites images (d_i) et (d_j) sont confondues. Cela signifie que le centre de la caméra se trouve sur une des tranches du trièdre. Excluons pour la suite ces cas dégénérés, d'où l'hypothèse de travail :

$$\forall i \neq j, \underline{\mathbf{h}}_i^T \underline{\mathbf{u}}_j \neq 0 \tag{3.17}$$

Sous cette hypothèse, le noyau du système (3.16) est :

$$\begin{pmatrix} \frac{\underline{\mathbf{h}}_3^T \underline{\mathbf{u}}_2}{h_2} \\ \frac{\underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_3}{h_3} \\ \frac{\underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_1}{h_1} \end{pmatrix}$$

Les ϵ_i vérifient donc :

$$\epsilon_1 = \lambda \frac{\underline{\mathbf{h}}_3^T \underline{\mathbf{u}}_2}{h_2} \quad \epsilon_2 = \lambda \frac{\underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_3}{h_3} \quad \epsilon_3 = \lambda \frac{\underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_1}{h_1}$$

Ceci nous fournit donc trois équations :

$$\begin{aligned}
(\underline{\mathbf{u}}_1 \times \underline{\mathbf{h}}_1)^T \underline{\mathbf{h}}_1^* &= \lambda \frac{\underline{\mathbf{h}}_3^T \underline{\mathbf{u}}_2}{h_2} \\
(\underline{\mathbf{u}}_2 \times \underline{\mathbf{h}}_2)^T \underline{\mathbf{h}}_2^* &= \lambda \frac{\underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_3}{h_3} \\
(\underline{\mathbf{u}}_3 \times \underline{\mathbf{h}}_3)^T \underline{\mathbf{h}}_3^* &= \lambda \frac{\underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_1}{h_1}
\end{aligned}$$

où l'on suppose connus les $\underline{\mathbf{h}}_i^*$ et les $\underline{\mathbf{u}}_i^*$, ainsi que les $\underline{\mathbf{u}}_i$ (puisqu'égaux aux $\underline{\mathbf{u}}_i^*$ après convergence). Quels sont alors les $\underline{\mathbf{h}}_i$ qui vérifient ces trois équations?

La première équation peut être transformée en :

$$-\underline{\mathbf{h}}_1^T (\underline{\mathbf{u}}_1 \times \underline{\mathbf{h}}_1^*) = \lambda \frac{\underline{\mathbf{h}}_3^T \underline{\mathbf{u}}_2}{h_2}$$

par permutation du produit mixte. Utilisons à présent l'orthogonalité du trièdre pour écrire $\underline{\mathbf{u}}_1 = \underline{\mathbf{u}}_2 \times \underline{\mathbf{u}}_3$ puis développons le double produit vectoriel ainsi obtenu pour aboutir à :

$$-\underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_1^* \ \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_1 + \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_1^* \ \underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_1 = \lambda \frac{\underline{\mathbf{h}}_3^T \underline{\mathbf{u}}_2}{h_2}$$

Similairement, on obtient pour les deux autres équations :

$$\begin{aligned}
-\underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_2^* \ \underline{\mathbf{u}}_1^T \underline{\mathbf{h}}_2 + \underline{\mathbf{u}}_1^T \underline{\mathbf{h}}_3^* \ \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_2 &= \lambda \frac{\underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_3}{h_3} \\
-\underline{\mathbf{u}}_1^T \underline{\mathbf{h}}_3^* \ \underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_3 + \underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_3^* \ \underline{\mathbf{u}}_1^T \underline{\mathbf{h}}_3 &= \lambda \frac{\underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_1}{h_1}
\end{aligned}$$

qui font apparaître les inconnues : $\underline{\mathbf{u}}_1^T \underline{\mathbf{h}}_2$, $\underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_2$, $\underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_3$, $\underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_3$, $\underline{\mathbf{u}}_1^T \underline{\mathbf{h}}_3$ et $\underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_1$.

Les trois droites du trièdre étant concourantes, elles vérifient

$$\forall i \neq j, h_i \underline{\mathbf{h}}_i^T \underline{\mathbf{u}}_j + h_j \underline{\mathbf{h}}_j^T \underline{\mathbf{u}}_j = 0$$

Asservissement visuel à partir de droites

grâce à quoi l'on peut réduire les inconnues aux seules : $\underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_2$, $\underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_3$, $\underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_3$ et obtenir le système homogène suivant :

$$\begin{pmatrix} \frac{\lambda}{h_2} & \frac{h_1 h_2^*}{h_3 h_3^*} \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_2^* & -\frac{h_2 h_1^*}{h_3 h_3^*} \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_1^* \\ \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_1^* & -\underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_1^* & \frac{\lambda}{h_3} \\ \frac{h_1}{h_2} \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_2^* & -\frac{\lambda}{h_3} & -\frac{h_1^*}{h_2^*} \underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_1^* \end{pmatrix} \begin{pmatrix} \underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_2 \\ \underline{\mathbf{h}}_1^T \underline{\mathbf{u}}_3 \\ \underline{\mathbf{h}}_2^T \underline{\mathbf{u}}_3 \end{pmatrix} = 0$$

dont le déterminant est :

$$\lambda \left(h_3^* (\underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_1^*)^2 h_1^* h_3^2 + \lambda^2 h_3^* h_2^* + (\underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_1^*)^2 h_1^* h_2^2 h_2^* + h_1^2 (h_2^*)^2 (\underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_2^*)^2 \right)$$

Rappelons que les h_i et les h_i^* sont strictement positifs puisque nous avons supposé que les droites ne passaient jamais par le centre de la caméra. Ce déterminant est donc positif et ne s'annule que lorsque $\lambda = 0$ ou $\underline{\mathbf{u}}_2^T \underline{\mathbf{h}}_1^* = \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_1^* = \underline{\mathbf{u}}_3^T \underline{\mathbf{h}}_2^* = 0$.

Cette dernière condition est impossible à moins que la position désirée du trièdre soit telle que l'une des droites passe par le centre de la caméra. Quant au cas $\lambda = 0$, il correspond au cas où tous les ϵ_i sont nuls, c.-à-d. que la convergence est atteinte.

Enfin, hormis ces cas, le système est homogène et inversible, donc la solution est le vecteur nul. Or, ceci est incompatible avec l'hypothèse (3.17).

Par conséquent, $\sum_{i=1}^{3} \epsilon_i \underline{\mathbf{h}}_i = 0 \Rightarrow \epsilon_i = 0, i = 1..3$.

2. De manière similaire, et parce que les h_i sont strictement positifs, on prouve que $\sum_{i=1}^{3} \frac{\epsilon_i \underline{\mathbf{h}}_i}{h_i} = 0$ implique aussi la nullité des ϵ_i. □

En conclusion, les deux commandes proposées permettent l'obtention de la pose partielle et leurs domaines de convergence sont parfaitement définis. Il reste donc à définir une commande en profondeur le long de la droite de vue passant par le centre du trièdre.

Profondeur

Nous devons faire une commande en translation le long d'une direction constante en observant l'erreur $e_s = (s - s^*)$ sur le positionnement de la tache laser le long de la droite épipolaire laser. Comme la direction est maintenue constante par la régulation de la pose partielle proposée ci-dessus, on peut utiliser la formule (3.8) qui découle de la remarque faite en page 83. Elle se traduit ici par :

$$\begin{aligned} \mathbf{V} &= v \underline{\mathbf{u}}_c \\ \dot{s} &= k \frac{v}{z^2} \end{aligned}$$

où $\underline{\mathbf{u}}_c$ est la direction, maintenue constante, de la droite de vue passant par le centre du trièdre ; v est la valeur algébrique de la vitesse $\mathbf{V}$; k est un scalaire dont le signe dépend de l'orientation relative du faisceau laser et du plan qu'il illumine ainsi que du choix du sens de $\underline{\mathbf{u}}_c$; enfin, z est la distance, inconnue, entre l'origine du faisceau laser et le point laser. On a donc

$$\begin{aligned} e_s &= (s - s^*) \\ \dot{e}_s &= k\frac{v}{z^2} \end{aligned}$$

Si l'on suppose toutes les quantités connues, on peut choisir le comportement exponentiel $\dot{e}_s = -\lambda e_s$ où $\lambda > 0$. On a alors l'expression idéale qui assure la stabilité asymptotique de la commande en profondeur :

$$v = -\frac{\lambda z^2}{k}(s - s^*)$$

Or, la profondeur z est inconnue, ce qui empêche d'utiliser une telle loi en pratique. Toutefois, puisque la profondeur n'intervient que par son carré, la fixer à une valeur constante et arbitraire ne changera pas la propriété de stabilité. En revanche, cela modifiera la vitesse de convergence et l'on n'aura plus forcément un comportement exponentiel.

La commande en profondeur est donc :

$$\mathbf{V} = -\frac{\mu_s}{k}(s - s^*)\underline{\mathbf{u}}_c, \quad \mu_s > 0 \tag{3.18}$$

Commande complète

Commande non projetée – On ajoute la commande en profondeur à la commande de la pose partielle en utilisant le formalisme des tâches redondantes [SLBE91]. En effet, on ne souhaite pas que la tâche laser s'oppose à la commande CPN (Figure 3.5), ce qui pourrait rendre instable la commande complète.

Par conséquent, il faut que la tâche laser n'influe pas sur la commande CPN, donc qu'elle se trouve dans le noyau de cette dernière. Comme la tâche laser est une translation pure, que nous noterons $\mathbf{V}_s$, nous devons nous concentrer sur le noyau de la partie translation de la commande CPN, que nous noterons $\mathbf{V}_h$.

On remarque que $\mathbf{V}_s$ et $\mathbf{V}_h$ ne sont pas orthogonales. En effet, $\mathbf{V}_s$ est parallèle à $\underline{\mathbf{u}}_c$ et le produit scalaire $\mathbf{V}_h^T\underline{\mathbf{u}}_c$ vaut :

$$\mathbf{V}_h^T\underline{\mathbf{u}}_c = -\mu_h \sum_{i=1}^{3}[\underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*)]^T\underline{\mathbf{u}}_c$$

Asservissement visuel à partir de droites

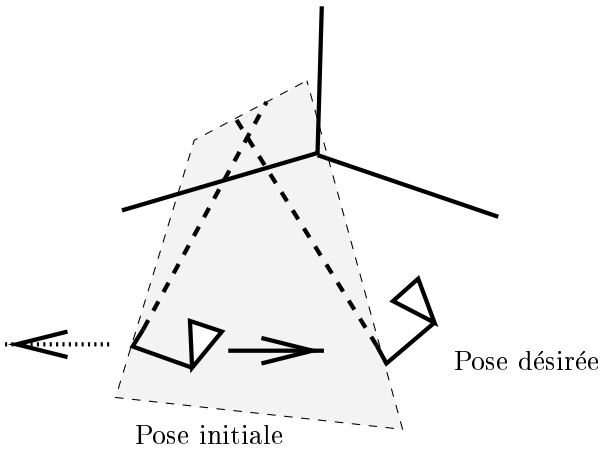


FIG. 3.5: *Un exemple d'incompatibilité: l'asservissement de la tache laser peut demander un mouvement global vers la gauche et la commande CPN souhaite un mouvement global vers la droite.*

Or, $\underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*) = (1 - \underline{\mathbf{h}}_i^T \underline{\mathbf{h}}_i^*)(\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i) + [(\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i)^T \underline{\mathbf{h}}_i^*]\underline{\mathbf{h}}_i$, d'où

$$\mathbf{V}_h^T \underline{\mathbf{u}}_c = -\mu_h \sum_{i=1}^{3} (1 - \underline{\mathbf{h}}_i^T \underline{\mathbf{h}}_i^*)(\underline{\mathbf{u}}_i \times \underline{\mathbf{h}}_i)^T \underline{\mathbf{u}}_c$$

De plus, en utilisant le fait que $\underline{\mathbf{h}}_i = \frac{\underline{\mathbf{u}}_c \times \underline{\mathbf{u}}_i}{\|\underline{\mathbf{u}}_c \times \underline{\mathbf{u}}_i\|}$ (obtenu par l'appartenance de la droite de vue passant par le centre du trièdre au plan d'interprétation défini par $\underline{\mathbf{h}}_i$), on aboutit à:

$$\mathbf{V}_h^T \underline{\mathbf{u}}_c = -\mu_h \sum_{i=1}^{3} (1 - \underline{\mathbf{h}}_i^T \underline{\mathbf{h}}_i^*) \frac{1 - (\underline{\mathbf{u}}_i^T \underline{\mathbf{u}}_c)^2}{\|\underline{\mathbf{u}}_c \times \underline{\mathbf{u}}_i\|}$$

qui n'est pas nul, hors cas invraisemblables en pratique où certains des couples $(\underline{\mathbf{h}}_i, \underline{\mathbf{h}}_i^*)$ forment un angle obtu et les autres non.

La commande complète obtenue dans le cas de la commande CPN non projetée est donc:

$$\begin{aligned} \mathbf{\Omega} &= \mu_u \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^* \\ \mathbf{V} &= \mathbf{V}_h + (\mathbf{I}_3 - \frac{\mathbf{V}_h \mathbf{V}_h^T}{\|\mathbf{V}_h\|^2}) \mathbf{V}_s \end{aligned}$$

En pratique, il faut cependant se limiter à n'activer la commande en profondeur qu'une fois la pose partielle atteinte, car la commande que nous avons définie n'est

valide qu'à pose partielle fixe. Dans ce cas, $\mathbf{V}_h = 0$ et l'opérateur de projection est réduit à l'identité et la commande complète, en séquence, devient :

$$\begin{aligned} \mathbf{\Omega} &= \mu_u \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^* \\ \mathbf{V} &= -\mu_h \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times (\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*) - \frac{\mu_s}{k}(s - s^*)\underline{\mathbf{u}}_c \end{aligned}$$

où les gains (μ_u, μ_h, μ_s), choisis par l'utilisateur, vérifient :

$$\mu_u > 0 \qquad \begin{cases} \mu_h = 0 & si\ \|\mathbf{\Omega}\| \leq \epsilon \\ \mu_h > 0 & sinon \end{cases} \qquad \begin{cases} \mu_s = 0 & si\ \|\mathbf{\Omega}\| \leq \epsilon\ et\ \|\mathbf{V}\| \leq \epsilon' \\ \mu_s > 0 & sinon \end{cases}$$

Commande non projetée – Dans ce cas, tout se simplifie puisque la tâche laser est trivialement orthogonale à la translation de la commande CPN projetée. En effet, cette dernière est une combinaison linéaire des $\underline{\mathbf{h}}_i$, tous orthogonaux à $\underline{\mathbf{u}}_c$ du fait de la concourance des trois droites. On notera que ce résultat reste valable lorsque le trièdre n'est pas orthogonal et lorsqu'il y a plus de trois droites dans le faisceau.

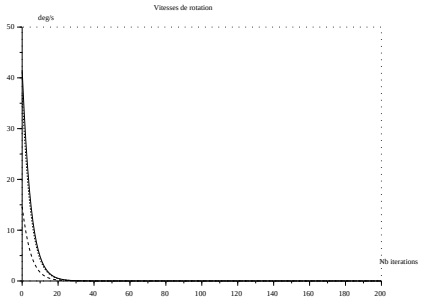
Par conséquent, la commande complète est tout simplement la somme de la commande CPN projetée et de la tâche laser :

$$\begin{aligned} \mathbf{\Omega} &= \mu_u \sum_{i=1}^{3} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^* \\ \mathbf{V} &= -\mu_h \sum_{i=1}^{3} \epsilon_i \underline{\mathbf{h}}_i - \frac{\mu_s}{k}(s - s^*)\underline{\mathbf{u}}_c \end{aligned}$$

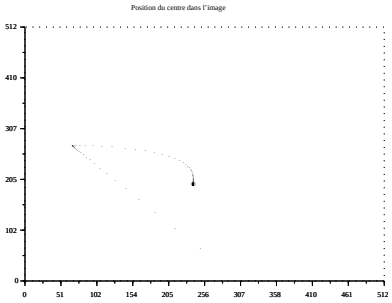
3.3 Simulation

Nous présentons ici quelques résultats de simulation afin de montrer le comportement qualitatif de la commande CPN complétée par l'asservissement de la tache laser. Nous avons choisi de comparer le comportement de la commande non projetée en séquence (Figure 3.6) ou simultanée (Figure 3.7), c.-à-d. lorsque la phase de translation est activée simultanément à celle de rotation, ainsi que la commande projetée simultanée (Figure 3.8). L'asservissement de la tache laser a, quant à lui et dans les trois cas, été activé après obtention de la pose partielle.

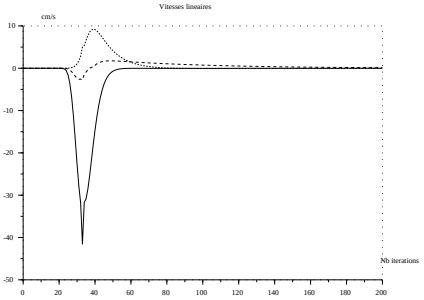
Commençons par étudier ces résultats dans le cas de la commande non projetée en séquence. La première commutation (rotation/translation) est bien visible sur les graphes des vitesses articulaires (Figures 3.6(a) et 3.6(c)). Elle se traduit par une convergence exponentielle de l'erreur en orientation accompagnée d'une



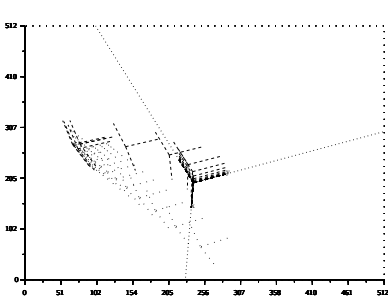
(a) Rotation



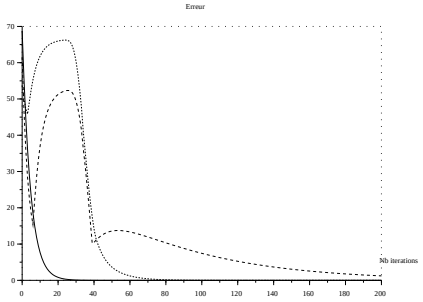
(b) Centre



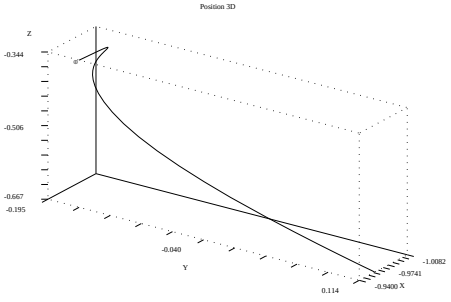
(c) Translation



(d) Image du trièdre

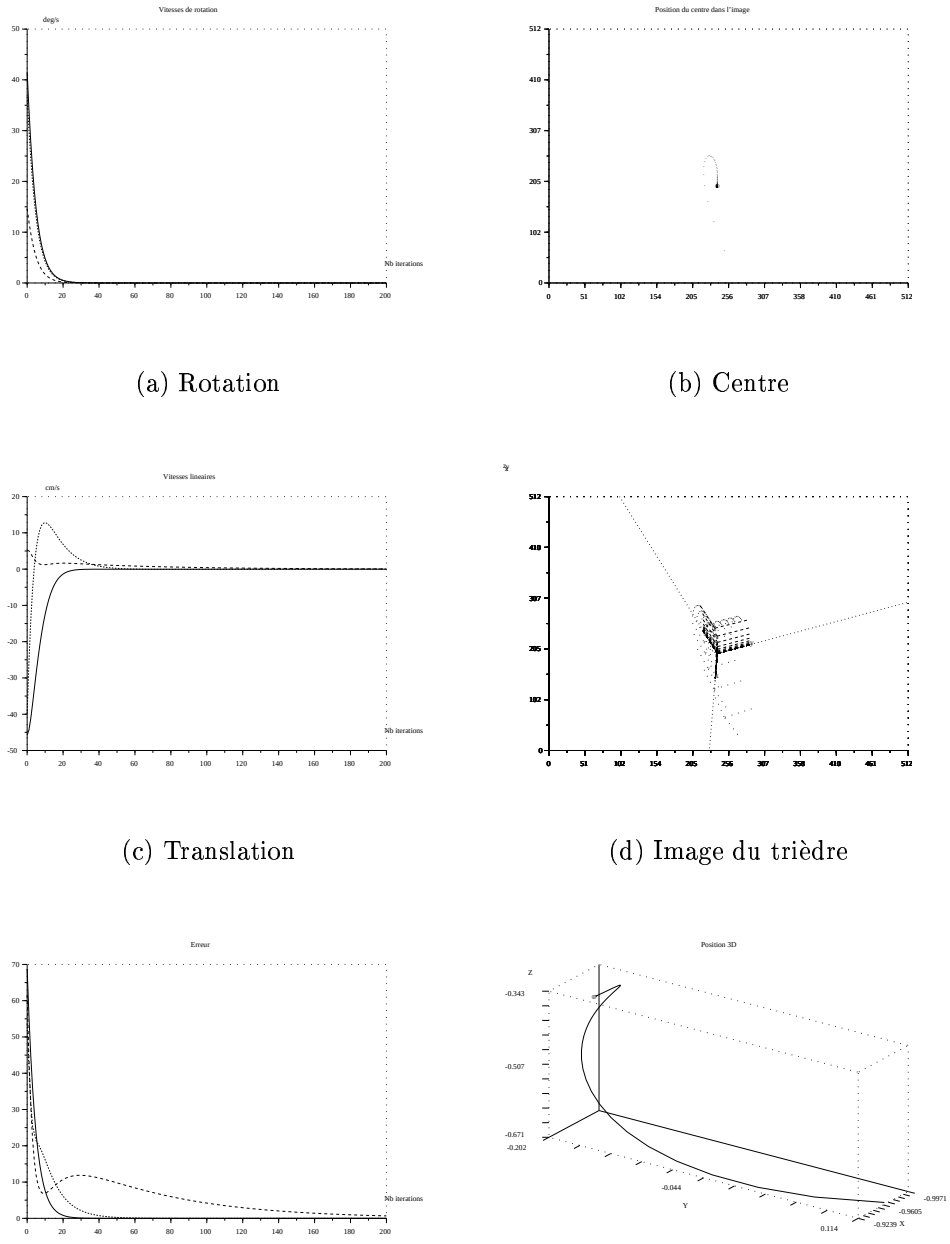


(e) Erreurs en orientation (trait plein), sur les droites image (pointillés) et sur la tache laser (tirets)



(f) Trajectoire dans l'espace

FIG. 3.6: *Simulation du comportement de la commande non projetée en séquence*



(a) Rotation

(b) Centre

(c) Translation

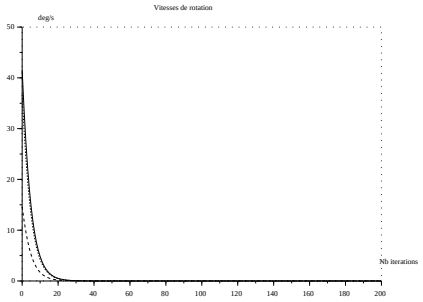
(d) Image du trièdre

(e) Erreurs

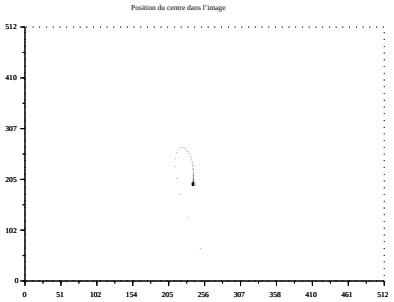
(f) Trajectoire dans l'espace

FIG. 3.7: *Simulation du comportement de la commande non projetée simultanée*

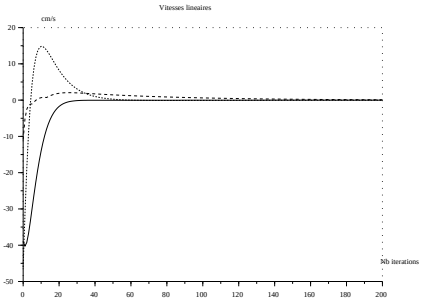
Asservissement visuel à partir de droites



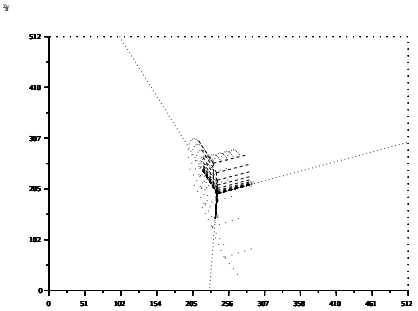
(a) Rotation



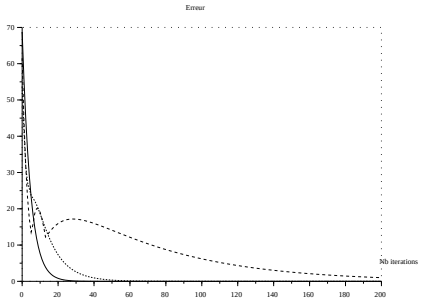
(b) Centre



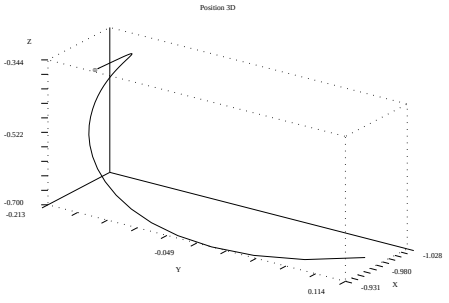
(c) Translation



(d) Image du trièdre



(e) Erreurs



(f) Trajectoire dans l'espace

FIG. 3.8: *Simulation du comportement de la commande projetée simultanée*

augmentation des erreurs sur les droites image et sur la tache laser puis, dans un deuxième temps, par la diminution des erreurs sur les droites image (Figure 3.6(e)). Ce comportement dans l'image est bien visible sur le tracé des projections du trièdre (Figure 3.6(d)) et de son centre (Figure 3.6(b)) : un déplacement continu vers la gauche, puis à l'activation de la translation un déplacement vers la position désirée dans l'image (grand trièdre). L'activation de l'asservissement sur la tache laser se voit bien sur la trajectoire du centre du repère de la caméra dans l'espace (Figure 3.6(f)) : un mouvement « arrondi » correspondant à l'obtention de la pose partielle par translation suivi d'un bout de trajectoire rectiligne. Cette activation se traduit par la convergence de l'erreur sur la tache laser, mais est en revanche invisible dans l'image et se distingue mal sur la courbe des vitesses.

Lorsque l'on active la partie translation de la commande non projetée en même temps que sa partie rotation, cela se voit dans les courbes de vitesses (Figures 3.7(a) et 3.7(c)). Le découplage partiel de la commande laisse inchangée la convergence en orientation, mais supprime la dérive des erreurs sur les droites image et la tache laser (Figure 3.7(e)). Cela se traduit par une moins grande amplitude du mouvement du trièdre dans l'image (Figures 3.7(b) et 3.7(d)), mais une plus grande amplitude dans l'espace (Figure 3.7(f)). Enfin, la convergence des droites dans l'image étant atteinte plus tôt, la commande laser est aussi activée plus tôt (Figure 3.7(e)).

La différence entre la commande projetée simultanée et la commande non projetée simultanée est un peu moins visible. En effet, les courbes des vitesses de cette dernière (Figures 3.8(a) et 3.8(c)) ressemblent à celles de la première (Figures 3.7(a) et 3.7(c)). Les trajectoires dans l'image ne se distinguent guère (Figures 3.7(b) 3.7(d), 3.8(b), et 3.8(d)). Il faut se pencher sur la courbe des erreurs (Figure 3.8(e)) pour voir une convergence plus lente sur la tache laser dans le cas projetée, qui se traduit par une trajectoire dans l'espace, dont la première phase est raccourcie et la seconde, allongée (Figure 3.8(f)). Ceci traduit un moins grand débattement en profondeur de la commande projetée par rapport à la commande non projetée et signifie que la commande en profondeur se fait en utilisant plus les informations laser.

En conclusion, il semble donc qu'il vaille mieux utiliser la commande projetée simultanée. Elle doit être simultanée pour réduire les risques de sortie de l'image et projetée pour diminuer ceux de collision avec le trièdre.

3.4 Application à un cas concret

3.4.1 Contexte industriel

Dans le cadre du projet VIGOR [VIG01], nous avons collaboré avec un chantier naval, dont l'objectif est d'automatiser la soudure de pièces de bateau. Jusqu'à présent, des modèles CAO étaient abondamment utilisés mais sans réel succès puisque

FIG. 3.9: *Le type de pièce qu'il faut traiter*

la taille des pièces mises en jeu ne manque pas d'induire des variations importantes par rapport aux modèles.

Dans 75% des cas, les pièces à assembler sont constituées de 3 plaques perpendiculaires (Figure 3.9). Un scénario de soudage est disponible, qui démarre par le positionnement de la torche de soudage au centre de ce trièdre orthogonal. Sur cette constatation, une deuxième approche, plus interactive, a été testée par le chantier naval. Elle consiste à trouver la position du trièdre par contacts : le robot se déplace lentement jusqu'à ce qu'un capteur de force lui indique qu'il a touché une face, on recommence alors l'opération pour la face suivante. Cette opération souffre de deux inconvénients majeurs. Elle est lente, puisque les contacts ne doivent pas intervenir à grande vitesse. Et, surtout, elle est suivie d'un positionnement de la torche en boucle ouverte (autant dire, « en aveugle ») sur la base de ces contacts.

La solution proposée dans ce chapitre permet donc de remplacer cette opération lente et ce positionnement en boucle ouverte par un positionnement rapide en boucle ouverte.

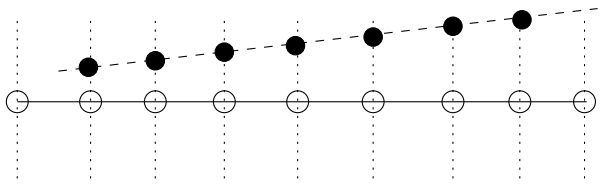
3.4.2 Traitement de l'image

Suivi du trièdre

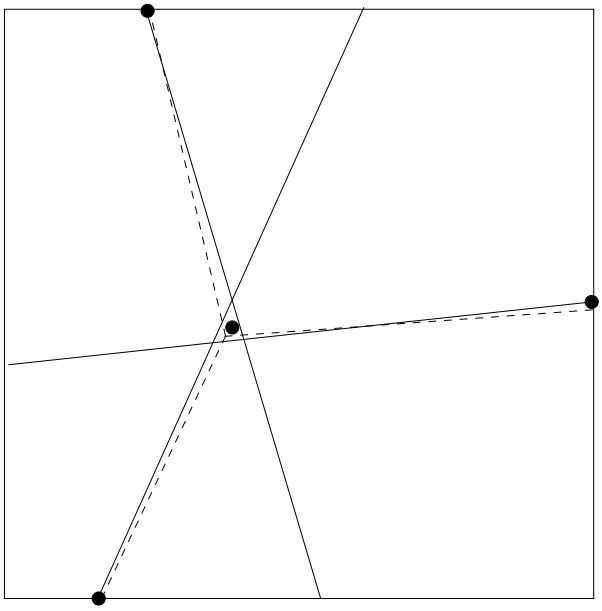
Pour extraire les droites des images, nous avons fait appel aux compétences de l'IRISA à Rennes. Ainsi, Éric Marchand nous a-t-il fourni un « suiveur de trièdre » qui fonctionne en 3 phases :

1° Sur chaque segment du trièdre détecté dans l'image précédente, on choisit un certain nombre de *sites* à partir desquels on cherche selon la perpendiculaire à la droite le pixel le plus corrélé avec le pixel de départ (Figure 3.10(a)). Puis, l'on estime la droite qui porte ces maxima ;

2° On estime le centre du trièdre et l'intersection de la partie visible des droites avec les bords de l'image (Figure 3.10(b)) ;



(a) À partir de sites (cercles blancs) situés sur le droite détectée dans l'image précédente (trait plein), on estime la position actuelle de la droite (pointillés) en cherchant les maxima de corrélation (points noirs)



(b) À partir des estimations des trois droites (trait plein), on estime leur intersection commune et les extrémités du trièdre (points noirs). En pointillés, le trièdre recherché.

FIG. 3.10: *Principe du suivi du trièdre dans l'image*

3° On optimise ces valeurs par recalage sur les variations de gradient de l'image. Des démonstrations de ce logiciel sont disponibles, au jour de l'écriture de ce chapitre, sur le serveur de l'IRISA : `http://www.irisa.fr/AVEC/`.

Calcul des orientations

Pour calculer les orientations des droites, nous nous sommes inspirés de la méthode proposée de calcul de pose à partir de 4 points [HCLL89]. Pour cela, nous avons tiré parti du fait que le trièdre est constitué de 3 demi-droites physiques. Ces dernières ont donc été orientées de leur intersection commune vers leur extrémité à l'infini. Leurs projections ont été orientées de la même manière. Alors, si l'on note $\underline{\mathbf{u}}_c$ le vecteur directeur de la droite de vue passant par le centre du trièdre et $\underline{\mathbf{v}}_i = \underline{\mathbf{u}}_c \times \underline{\mathbf{u}}_i$, on a :

$$\underline{\mathbf{u}}_i = \cos\theta_i \underline{\mathbf{u}}_c + \sin\theta_i \underline{\mathbf{v}}_i, \quad i = 1..3, \quad \sin\theta_i > 0$$

En résolvant le système ainsi constitué, on aboutit à une solution de la forme

$$\begin{aligned} \cos\theta_i &= \pm\sqrt{\frac{\underline{\mathbf{v}}_{i-1}^T \underline{\mathbf{v}}_{i+1}}{\underline{\mathbf{v}}_{i-1}^T \underline{\mathbf{v}}_i \underline{\mathbf{v}}_i^T \underline{\mathbf{v}}_{i+1} + \underline{\mathbf{v}}_{i-1}^T \underline{\mathbf{v}}_{i+1}}} \\ \sin\theta_i &= \sqrt{\frac{\underline{\mathbf{v}}_{i-1}^T \underline{\mathbf{v}}_i \underline{\mathbf{v}}_i^T \underline{\mathbf{v}}_{i+1}}{\underline{\mathbf{v}}_{i-1}^T \underline{\mathbf{v}}_i \underline{\mathbf{v}}_i^T \underline{\mathbf{v}}_{i+1} + \underline{\mathbf{v}}_{i-1}^T \underline{\mathbf{v}}_{i+1}}} \end{aligned}$$

Il est entendu que les calculs d'indice se font dans $\mathbb{Z}/3\mathbb{Z}$. Enfin, le signe des $\cos\theta_i$ dépend du choix de la solution, concave ou convexe, et est identique dans le cas non dégénéré où les trois demi-droites sont effectivement visibles.

3.4.3 Résultats expérimentaux

Validation de la commande CPN

Pour valider expérimentalement le bien-fondé de la commande CPN, nous avons annulé la commande en profondeur pour supprimer toute interférence. De plus, nous avons travaillé sur une maquette de trièdre, plutôt que sur la pièce de bateau. En effet, cette dernière ne présente que peu de contraste aux jonctions des plaques et, à l'inverse, contient des points de soudure et autres taches de fort contraste. Ces caractéristiques intrinsèques à la pièce de bateau ralentissent la tâche de suivi et le rendent instable.

Cette série d'expérience a été menée sur le robot cartésien de l'IRISA sur l'effecteur duquel une caméra est montée. Du fait de ce dispositif, nous avons utilisé la commande CPN projetée avec activation simultanée des commandes en rotation et en translation. On sort donc des conditions suffisantes de convergence, pourtant

nous allons voir qu'en pratique, notre intuition concernant la possibilité d'étendre les résultats de la commande en séquence au cas de la commande simultanée semble valide.

Dans toutes les expériences présentées ici, la consigne a été apprise en plaçant le robot dans une position finale désirée.

Translation pure – Dans une première expérience, nous avons écarté la caméra d'une dizaine de centimètres de sa position de consigne (Figure 3.11). On remarquera, tout d'abord, qu'au cours de l'accomplissement de la tâche, une brève erreur de suivi s'est produite aux environs de la centième itération (Figure 3.12, gauche), ce qui explique les quelques aberrations des courbes suivantes. On notera que le calcul des orientations 3D est stable (Figure 3.12, droite), puisque l'erreur de suivi le perturbe peu. De plus, ces orientations restent constantes car le mouvement effectué est une translation pure : la commande en rotation est nulle (Figure 3.13, droite) au bruit près. Ceci est conforme au fait que le mouvement désiré entre la position initiale et la position finale est une translation pure et met bien en évidence le découplage partiel de la commande CPN. La translation a, dans cette expérience, un comportement exponentiel décroissant (Figure 3.13, gauche). Conformément à nos résultats théoriques, l'erreur dans l'image ($\sum_{i=1}^{3} \|\underline{\mathbf{h}}_i - \underline{\mathbf{h}}_i^*\|$) décroît et l'erreur en orientation est nulle au bruit près (Figure 3.14, gauche). On notera enfin la trajectoire quasiment rectiligne du centre du trièdre dans l'image (Figure 3.14, droite).

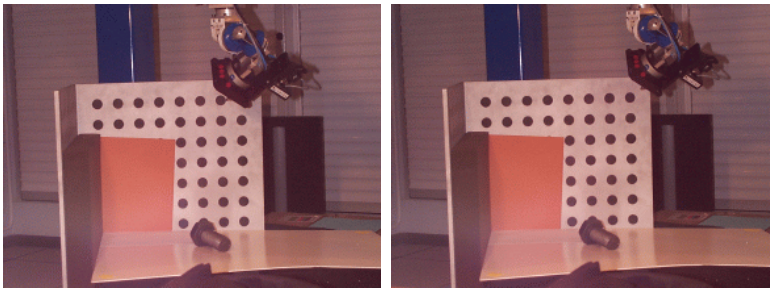


FIG. 3.11: *À gauche : position de consigne du robot, à droite : position initiale du robot.*

Rotation pure – Dans les mêmes conditions expérimentales, nous avons déplacé la caméra depuis sa position de consigne par une rotation de 100 degrés autour de l'axe optique. On constate une bonne stabilité d'extraction des droites de l'image (Figure 3.15, gauche) et du calcul d'orientation (Figure 3.15, droite). La commande en orientation se fait, comme souhaité, principalement autour de l'axe optique (Figure 3.16, droite) et induit une commande en translation asymptotiquement stable (Figure 3.16, gauche) qui permet de garder le trièdre dans l'image (Figure 3.17, droite).

Asservissement visuel à partir de droites

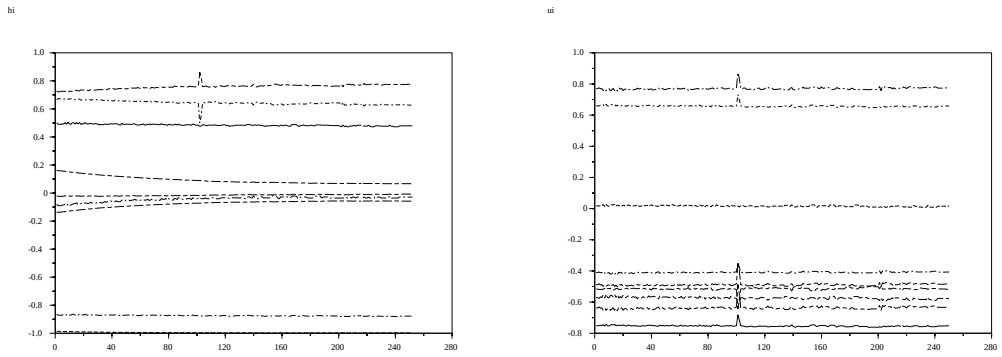


FIG. 3.12: *Translation pure : Évolution des coefficients des vecteurs* $\underline{\mathbf{h}}_i$ *(gauche) et de ceux des vecteurs* $\underline{\mathbf{u}}_i$ *(droite) au cours de l'asservissement.*

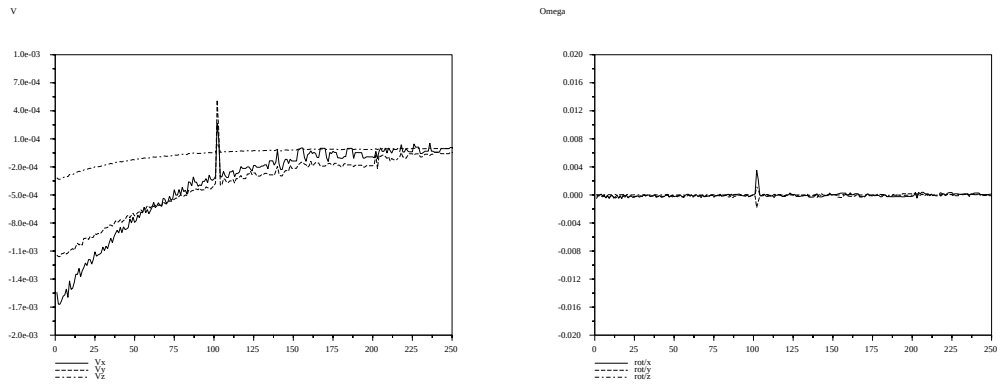


FIG. 3.13: *Translation pure : Évolution de la commande en translation (en* m/s*) à gauche et rotation (en* rad/s*) à droite.*

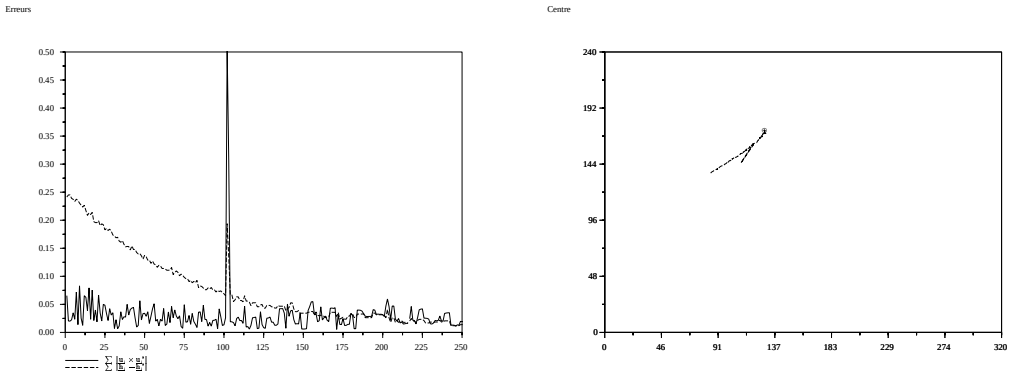


FIG. 3.14: *Translation pure : Évolution des erreurs (gauche) et trajectoire du centre du trièdre dans l'image (droite).*

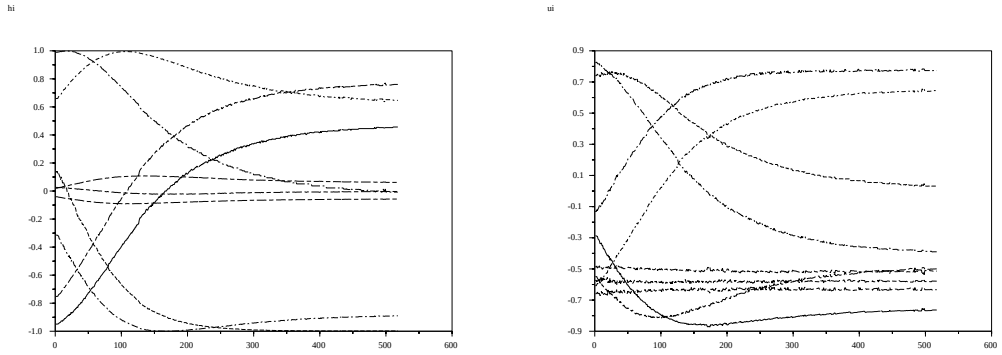


FIG. 3.15: *Rotation pure : Évolution des coefficients des vecteurs* $\underline{\mathbf{h}}_i$ *(gauche) et de ceux des vecteurs* $\underline{\mathbf{u}}_i$ *(droite) au cours de l'asservissement.*

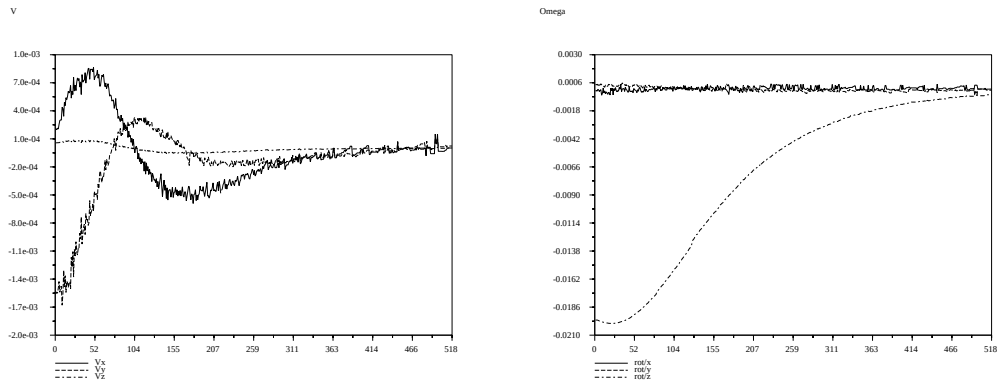


FIG. 3.16: *Rotation pure : Évolution de la commande en translation (en* m/s*) à gauche et rotation (en* rad/s*) à droite.*

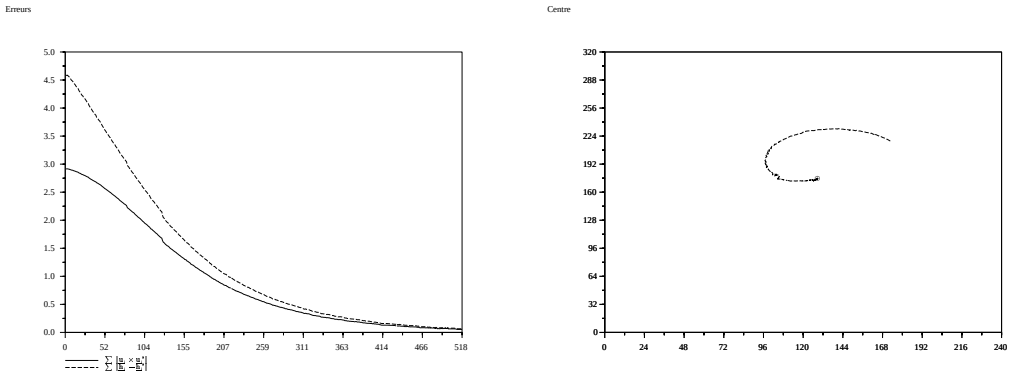


FIG. 3.17: *Rotation pure : Évolution des erreurs (gauche) et trajectoire du centre du trièdre dans l'image (droite).*

Asservissement visuel à partir de droites

On notera que la commande en rotation ne converge pas exponentiellement mais débute par une légère accélération (Figure 3.16, droite). Néanmoins, aussi bien l'erreur en orientation que l'erreur dans l'image décroissent (Figure 3.17, gauche). Ceci est en fait cohérent avec le Lemme 1, page 51. En effet, l'erreur en rotation initiale est supérieure à $\frac{\pi}{2}$. Par conséquent, la norme de $\sum_{i=1}^{3} \underline{\mathbf{u}}_i \times \underline{\mathbf{u}}_i^*$, proportionnelle au sinus de l'angle de la rotation résiduelle entre l'orientation courante et l'orientation désirée, va croître jusqu'à ce que l'angle résiduel atteigne, en décroissant, $\frac{\pi}{2}$ puis convergera exponentiellement par la suite.

Mouvement complexe – Nous terminons cette série d'expérience par un mouvement complexe, au sens qu'il comporte une translation d'une cinquantaine de centimètres et une rotation d'environ 80 degrés (Figure 3.18). Il est complexe à un deuxième titre puisque nous avons placé la caméra dans une position initiale proche d'un des bords du trièdre. Cette situation est difficile pour le suiveur puisque deux des droites 3D sont proches de se projeter en une même droite image et risquent d'être confondues. Elle est aussi délicate pour la commande puisque la caméra se trouve à proximité d'une singularité (§ 3.2.3).

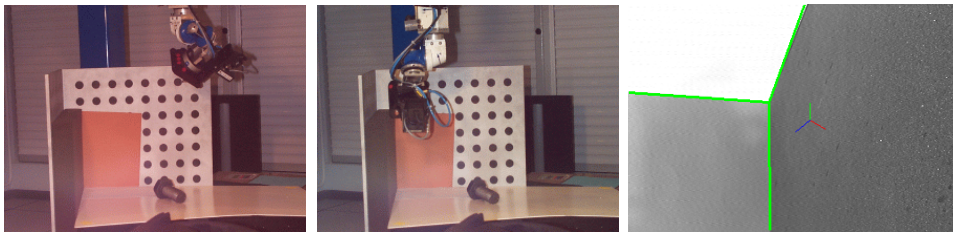


FIG. 3.18: *À gauche : position de consigne du robot, au centre : position initiale du robot, à droite : image initiale. N.B. : La consigne dans l'image est représentée par le petit trièdre tricolore.*

On constate que la commande écarte bien la caméra de cette singularité (Figure 3.19, gauche) : les coefficients des vecteurs $\underline{\mathbf{h}}_i$ qui étaient voisins à l'instant initial s'éloignent les uns des autres, alors qu'ils seraient confondus en la singularité. La commande en rotation est désormais strictement décroissante (Figure 3.20, droite) puisque l'angle résiduel entre les orientations initiale et désirée est inférieur à $\frac{\pi}{2}$. La commande en translation quant à elle n'est qu'asymptotiquement stable (Figure 3.20, gauche), car perturbée par la commande en rotation. En revanche, les erreurs décroissent (Figure 3.21, gauche). Enfin, la trajectoire dans l'image du centre du trièdre n'est plus rectiligne (Figure 3.21, droite) et dépend du réglage relatif des gains μ_u et μ_h. En effet, cette trajectoire a été obtenue avec $\mu_u = 0,008$ et $\mu_h = 0.01$ alors que le choix $\mu_u = \mu_h = 0.01$ ne permettait pas de garder le trièdre dans l'image.

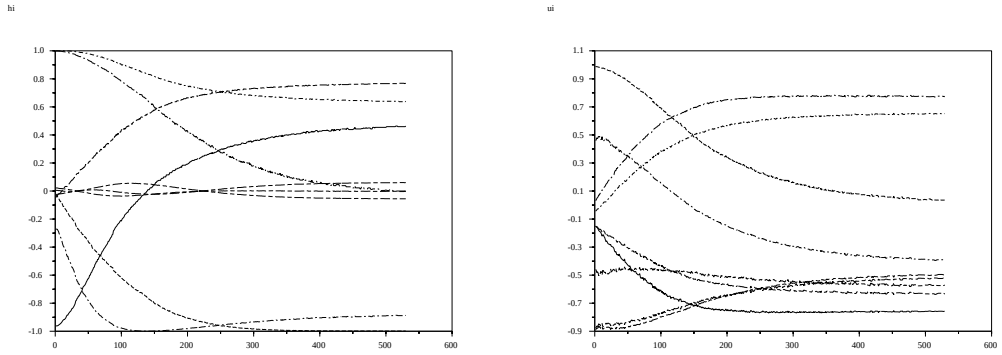


FIG. 3.19: *Mouvement complexe : Évolution des coefficients des vecteurs* $\underline{\mathbf{h}}_i$ *(gauche) et de ceux des vecteurs* $\underline{\mathbf{u}}_i$ *(droite) au cours de l'asservissement.*

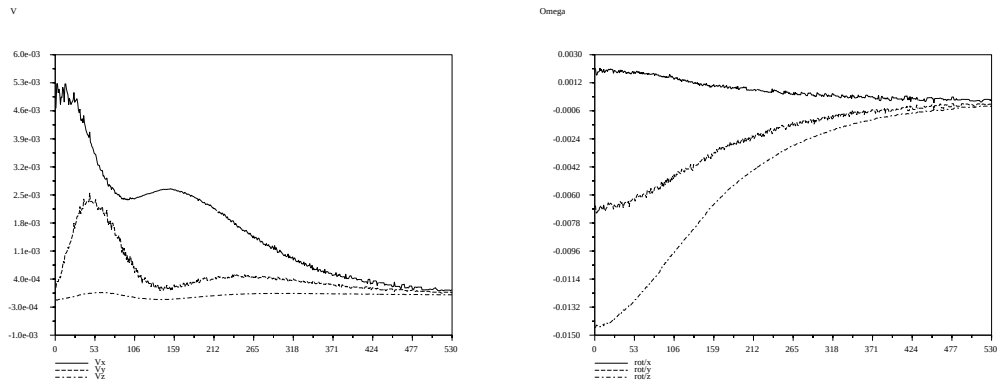


FIG. 3.20: *Mouvement complexe : Évolution de la commande en translation (en m/s) à gauche et rotation (en rad/s) à droite.*

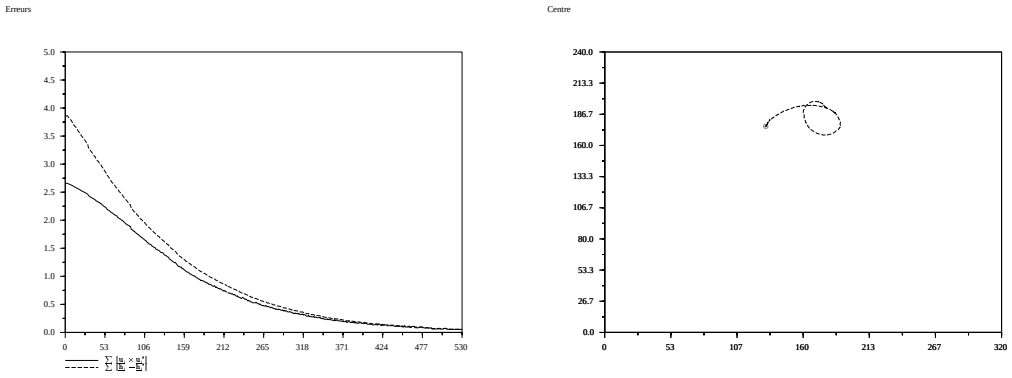


FIG. 3.21: *Mouvement complexe : Évolution des erreurs (gauche) et trajectoire du centre du trièdre dans l'image (droite).*

Asservissement visuel à partir de droites

3.5 Conclusion

Dans ce chapitre, nous avons apporté une solution au positionnement par rapport à un trièdre orthogonal. Pour cela, nous avons appliqué la commande CPN dont nous avons pu affiner, dans ce cas, les propriétés (stabilité asymptotique, singularités). La distance au trièdre n'étant pas observable à partir des seules droites extraites, la commande CPN ne permet d'obtenir qu'une pose partielle. Pour la compléter, nous avons proposé l'utilisation d'un faisceau laser non étalonné. Ainsi, nous pouvons commander la distance au trièdre par observation de la variation de la tache laser dans l'image sous la contrainte, maintenue par la commande CPN, que la pose partielle soit constante.

Cette étude théorique a été complétée par des simulations qui ont permis de qualifier le comportement de la commande proposée. Enfin, la commande CPN projetée a été validée expérimentalement dans le cas du trièdre. Les résultats expérimentaux semblent confirmer l'idée que les preuves de convergence obtenues pour une commande en séquence peuvent être étendues au cas de la commande simultanée. Il reste cependant plusieurs pistes expérimentales en suspens : comparaison avec les autres méthodes d'asservissement visuel à partir de droites, comparaison de la commande projetée et non projetée, ou encore, influence du choix des gains.

Auto-étalonnage pince-caméra

Rôti de Manjoux

Pour 8 personnes
Ingrédients : 1,5 kg d'aiguillette de rumsteck non bardée, 1 tomate, 1 oignon, 1 gousse d'ail.

- Piquer la viande d'ail.
- Faire blondir l'oignon émincé dans une cocotte. Le réserver.
- Saisir la viande de tous côtés.
- Remettre l'oignon. Couper la tomate en quartiers et l'ajouter dans la cocotte.
- Couvrir et laisser sur feu vif pendant 5 mn.
- Couper le gaz, couvrir d'un torchon 1/4 h–20 mn.
- Servir accompagné d'un gratin dauphinois (évidemment !) et de Mondeuse d'Arbin.

Chapitre 1

Étalonnage pince-caméra linéaire

1.1 Introduction

Dans ce chapitre, nous présentons une méthode linéaire d'étalonnage pince-caméra. La procédure pratique d'utilisation de cette méthode, représentée en Figure 1.1, est classique. À partir d'images d'une *mire de calibration*, c.-à-d. d'un objet manufacturé avec le plus grand soin, dont on connaît avec une très grande précision les caractéristiques, on calcule les poses successives de la caméra. Puis, à l'aide des mouvements préalablement enregistrés du robot, on peut calculer la transformation pince-caméra.

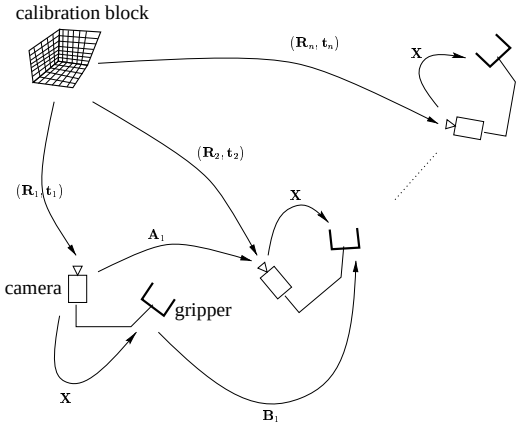


FIG. 1.1: *La procédure pratique d'étalonnage pince-caméra.*

Cette méthode est inspirée par la ressemblance du problème d'étalonnage pince-caméra avec l'équation matricielle dite de Sylvester. Elle a, notamment, pour but d'aider à l'analyse algébrique des mouvements nécessaires à l'étalonnage pince-caméra.

1.2 Formulation du problème

1.2.1 Construction du système linéaire

Pour faire apparaître la transformation pince-caméra, on peut considérer le déplacement d'un système constitué d'une pince et d'une caméra fixées l'une à l'autre (Figure 3.5, p. 22). Lorsque la pince se déplace selon la transformation rigide $\mathbf{B} = (\mathbf{R}_B, \mathbf{t}_B)$ (exprimée dans le repère pince), la caméra se déplace, elle, selon la transformation rigide $\mathbf{A} = (\mathbf{R}_A, \mathbf{t}_A)$ (exprimée dans le repère caméra). $\mathbf{A}$ et $\mathbf{B}$ sont alors liées par :

$$\mathbf{AX} = \mathbf{XB} \tag{1.1}$$

qui se décompose en :

$$\mathbf{R}_A\mathbf{R}_X - \mathbf{R}_X\mathbf{R}_B = \mathbf{0} \tag{1.2}$$

$$\mathbf{R}_X\mathbf{t}_B + \mathbf{t}_X - \mathbf{R}_A\mathbf{t}_X = \mathbf{t}_A \tag{1.3}$$

L'équation (1.2) peut s'écrire comme une équation de Sylvester, c'est-à-dire une équation matricielle de la forme :

$$\mathbf{UV} + \mathbf{VW} = \mathbf{T} \tag{1.4}$$

avec dans notre cas, $\mathbf{U} = \mathbf{R}_A$, $\mathbf{W} = -\mathbf{R}_B$ et $\mathbf{T} = \mathbf{0}$.

Les études classiques portant sur la résolution d'une telle équation [BS72, GNL79, RB89, HR92, DSH95, Dua96] mettent en évidence une condition nécessaire d'unicité de la solution : les matrices $\mathbf{U}$ et $\mathbf{W}$ ne doivent pas avoir de valeurs propres opposées. Ici, $\mathbf{U}$ est une matrice de rotation et a donc une valeur propre égale à 1 et $\mathbf{W}$ est le produit d'une matrice de rotation par -1 et a donc -1 pour valeur propre. La condition d'unicité n'est par conséquent pas vérifiée et ces méthodes ne sont pas applicables.

Cependant, certains auteurs [RB89, HR92, DSH95], formulent le problème d'une manière intéressante : en réordonnant les coefficients de la matrice $\mathbf{V}$ sous la forme d'un vecteur (une ligne après l'autre), on obtient un système équivalent à (1.4) de la forme :

$$(\mathbf{U} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{W})vec(\mathbf{V}) = vec(\mathbf{T}) \tag{1.5}$$

où

- *vec* est l'opérateur introduit par Neudecker [Neu69] transformant une matrice $\mathbf{M}$ de dimension $(m \times n)$ et de terme général M_{ij} en le vecteur

$$vec(\mathbf{M}) = (M_{11}, M_{12}, \ldots, M_{1n}, M_{21}, M_{22}, \ldots, M_{mn})^T$$

- le symbole $\otimes$ représente le produit de Kronecker[1] (voir [Bel60] pour les premiers théorèmes et [Bre78] pour une liste exhaustive de résultats théoriques) défini pour les matrices $\mathbf{M}$ et $\mathbf{N}$ de dimensions respectives $(m \times n)$ et $(o \times p)$ par la matrice de dimension $(mo \times np)$:

$$\mathbf{M} \otimes \mathbf{N} = \begin{pmatrix} M_{11}\mathbf{N} & \ldots & M_{1n}\mathbf{N} \\ \vdots & \ddots & \vdots \\ M_{m1}\mathbf{N} & \ldots & M_{mn}\mathbf{N} \end{pmatrix}$$

En s'inspirant de cette formulation, on obtient, après application de diverses propriétés du produit de Kronecker, un système équivalent au système formé par les équations (1.2) et (1.3), exprimé ici sous forme de blocs :

$$\left(\begin{array}{c|c} \mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B & \mathbf{0}_{9\times 3} \\ \hline \mathbf{I}_3 \otimes (\mathbf{t}_B^T) & \mathbf{I}_3 - \mathbf{R}_A \end{array}\right) \left(\begin{array}{c} vec(\mathbf{R}_X) \\ \hline \mathbf{t}_X \end{array}\right) = \left(\begin{array}{c} \mathbf{0}_{9\times 1} \\ \hline \mathbf{t}_A \end{array}\right) \tag{1.6}$$

Nous noterons par la suite $\mathbf{C}$, la matrice de ce système.

1.2.2 Étude de rang

Cas d'une translation pure

Dans le cas d'un mouvement de translation pure, on a la double égalité suivante :

$$\mathbf{R}_A = \mathbf{R}_B = \mathbf{I}$$

La matrice $\mathbf{C}$ du système (1.6) devient alors :

$$\mathbf{C}_{trans} = \left(\begin{array}{c|c} \mathbf{0}_{9\times 9} & \mathbf{0}_{9\times 3} \\ \hline \mathbf{I}_3 \otimes (\mathbf{t}_B^T) & \mathbf{0}_{3\times 3} \end{array}\right) \tag{1.7}$$

1. Cet opérateur est aussi appelé produit tensoriel.

Son rang est donc celui de la sous-matrice $\mathbf{I}_3 \otimes \mathbf{t}_B^T$ [2]. Cette matrice de dimension (3×9) s'écrit :

$$\mathbf{I}_3 \otimes \mathbf{t}_B^T = \begin{pmatrix} \mathbf{t}_B^T & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{t}_B^T & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \mathbf{t}_B^T \end{pmatrix}$$

qui est de rang nul si $\mathbf{t}_B = 0$ (cas inutile) et de rang 3 sinon.

De plus, on remarquera que dans le cas d'une translation pure, le système (1.2)-(1.3) se simplifie en :

$$\begin{aligned} \mathbf{R}_X &= \mathbf{R}_X \\ \mathbf{R}_X \mathbf{t}_B &= \mathbf{t}_A \end{aligned}$$

Cela est cohérent avec la simplification du système (1.7) en :

$$(\mathbf{I}_3 \otimes \mathbf{t}_B^T) vec(\mathbf{R_X}) = \mathbf{t}_A$$

et signifie qu'il n'est pas possible de retrouver $\mathbf{t}_X$ à partir de simples translations.

Cas de n translations pures Dans le cas de $n > 1$ translations pures, on peut empiler n systèmes simplifiés et obtenir le système de $3n$ équations à 9 inconnues :

$$\begin{pmatrix} \mathbf{I}_3 \otimes \mathbf{t}_{B_1}^T \\ \mathbf{I}_3 \otimes \mathbf{t}_{B_2}^T \\ \vdots \\ \mathbf{I}_3 \otimes \mathbf{t}_{B_n}^T \end{pmatrix} vec(\mathbf{R_X}) = \begin{pmatrix} \mathbf{t}_{A_1} \\ \mathbf{t}_{A_2} \\ \vdots \\ \mathbf{t}_{A_n} \end{pmatrix}$$

Ce système est de rang 9 si l'on dispose de 3 translations indépendantes.

Conclusion Dans le cas d'une translation pure (non nulle), le rang du système est de rang 3. La simplification du système met en évidence qu'avec 3 translations indépendantes, on peut retrouver $\mathbf{R}_X$, l'orientation relative de la pince vis-à-vis de la caméra. En revanche, il est impossible de retrouver la translation séparant la pince de la caméra. Ce résultat a aussi été trouvé par [Zhu98b].

Cas d'une rotation pure

Il faut d'abord se rappeler que lorsque deux repères rigidement liés se déplacent, ils correspondent au même vissage et possèdent donc 2 valeurs communes : ils ont le

2. On peut abandonner le parenthèsage car $(\mathbf{M} \otimes \mathbf{N})^T = (\mathbf{M}^T) \otimes (\mathbf{N}^T)$.

même angle de rotation et les produits scalaires de leur vecteur de translation avec leur axe de rotation respectifs sont égaux.

Il faut ensuite remarquer que les déplacements de la caméra et du robot ne peuvent pas être simultanément des rotations pures, à moins que leurs repères respectifs n'aient la même origine (ce qui signifierait alors que la translation pince-caméra $\mathbf{t}_X$ était nulle) ou que l'axe de la rotation du repère caméra ne soit parallèle à la translation pince-caméra.

Pour cela, réécrivons l'équation (1.3) :

$$\mathbf{t}_A = \mathbf{R}_x\mathbf{t}_B + (\mathbf{I} - \mathbf{R}_A)\mathbf{t}_X$$

Si nous mettons $\mathbf{t}_B = 0$ dans cette équation, nous obtenons :

$$\mathbf{t}_A = (\mathbf{I} - \mathbf{R}_A)\mathbf{t}_X$$

Ainsi, $\mathbf{t}_A$ ne peut être nul que dans les deux cas énoncés précédemment : $\mathbf{t}_X$ est l'axe de rotation de $\mathbf{R}_A$ ou $\mathbf{t}_X = 0$. En effet, le cas où $\mathbf{I} = \mathbf{R}_A$ ne peut se produire que lorsque $\mathbf{R}_B$ est aussi l'identité (en vertu de la première contrainte de rigidité), ce qui signifierait que ni le robot, ni la caméra n'ont bougé.

Inversement, si nous mettons $\mathbf{t}_A = 0$, nous obtenons :

$$\mathbf{t}_B = -\mathbf{R}_x^T(\mathbf{I} - \mathbf{R}_A)\mathbf{t}_X$$

qui s'annule dans les mêmes cas que précédemment.

Supposons donc que $\mathbf{t}_B = 0$, c'est-à-dire que l'on demande au robot d'effectuer une rotation pure. Dans ce cas, nous obtenons une autre expression pour $\mathbf{C}$:

$$\mathbf{C}_{rot} = \left(\begin{array}{c|c} \mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B & \mathbf{0}_{9\times 3} \\ \hline \mathbf{0}_{3\times 9} & \mathbf{I}_3 - \mathbf{R}_A \end{array}\right) \tag{1.8}$$

La matrice $\mathbf{C}_{rot}$ est donc diagonale par blocs. Son rang est donc :

$$rg(\mathbf{C}_{rot}) = rg(\mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B) + rg(\mathbf{I}_3 - \mathbf{R}_A)$$

Rang de $\mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B$ Calculer ce rang revient à étudier les valeurs propres de $\mathbf{R}_A \otimes \mathbf{R}_B$.

Soient $\lambda_i (i \in \{1,2,3\})$ (resp. $\mu_j (j \in \{1,2,3\})$), les 3 valeurs propres de $\mathbf{R}_A$ (resp. $\mathbf{R}_B$)). On sait que dans les 2 cas, ces valeurs propres sont $\{e^{i\theta}, e^{-i\theta}, 1\}$ où θ est l'angle commun de rotation.

De plus, une propriété du produit de Kronecker est que les valeurs propres du produit $\mathbf{A} \otimes \mathbf{B}$ sont les produits des valeurs propres de $\mathbf{A}$ par celles de $\mathbf{B}$ [Bel60].

Auto-étalonnage pince-caméra

Dans le cas présent, ces produits sont par conséquent :

$$\{1, 1, 1, e^{i\theta}, e^{i\theta}, e^{-i\theta}, e^{-i\theta}, e^{2i\theta}, e^{-2i\theta}\}$$

On distingue 3 cas :

- quand $\theta = 2k\pi$, k étant un entier quelconque, alors toutes les valeurs propres sont égales à 1 (ce qui est normal car alors les deux rotations sont nulles);
- quand $\theta = (2k+1)\pi$, il n'y a plus que 2 valeurs propres distinctes, 1 et -1, de multiplicités respectives 5 et 4;
- quand $\theta \neq k\pi$, la valeur propre 1 est de multiplicité 3.

Le rang de $\mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B$ est donc égal à 6 dans le cas général, 4 dans le cas d'une rotation de $(2k+1)\pi$ et 0 dans le cas dégénéré d'un mouvement nul. De plus, le rang de $\mathbf{I}_3 - \mathbf{R}_A$ est toujours 2, dans le cas d'une rotation pure non nulle.

En conclusion, en excluant le cas dégénéré et en considérant la mesure principale de l'angle de rotation θ (c.-à-d. ramenée dans l'intervalle $[0, 2\pi[$) :

$$\ker(\mathbf{C}_{rot}) = \underbrace{\ker(\mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B)}_{\begin{array}{l}=6 \text{ si } \theta \neq \pi \\ =4 \text{ sinon}\end{array}} + \underbrace{\ker(\mathbf{I}_3 - \mathbf{R}_A)}_{=2} = \begin{cases} =8 & \text{si } \theta \neq \pi \\ =6 & \text{sinon} \end{cases}$$

Cas de n rotations pures Dans le cas de $n > 1$ rotations pures du robot (c'est-à-dire que les $\mathbf{t}_{B_i}$ sont nuls), on peut former le système de $12n$ équations et 12 inconnues suivant :

$$\begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_{A_1} \otimes \mathbf{R}_{B_1} & \mathbf{0}_{9\times3} \\ \mathbf{0}_{3\times9} & \mathbf{I}_3 - \mathbf{R}_{A_1} \\ \mathbf{I}_9 - \mathbf{R}_{A_2} \otimes \mathbf{R}_{B_2} & \mathbf{0}_{9\times3} \\ \mathbf{0}_{3\times9} & \mathbf{I}_3 - \mathbf{R}_{A_2} \\ \vdots & \vdots \\ \mathbf{I}_9 - \mathbf{R}_{A_n} \otimes \mathbf{R}_{B_n} & \mathbf{0}_{9\times3} \\ \mathbf{0}_{3\times9} & \mathbf{I}_3 - \mathbf{R}_{A_n} \end{pmatrix} vec(\mathbf{X}) = \begin{pmatrix} \mathbf{0}_{9\times1} \\ \mathbf{t}_{A_1} \\ \mathbf{0}_{9\times1} \\ \mathbf{t}_{A_2} \\ \vdots \\ \mathbf{0}_{9\times1} \\ \mathbf{t}_{A_n} \end{pmatrix} \tag{1.9}$$

Ce système se découple en :

$$\begin{pmatrix} \mathbf{I}_3 - \mathbf{R}_{A_1} \\ \mathbf{I}_3 - \mathbf{R}_{A_2} \\ \vdots \\ \mathbf{I}_3 - \mathbf{R}_{A_n} \end{pmatrix} \mathbf{t}_X = \begin{pmatrix} \mathbf{t}_{A_1} \\ \mathbf{t}_{A_2} \\ \vdots \\ \mathbf{t}_{A_n} \end{pmatrix} \tag{1.10}$$

et

$$\begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_{A_1} \otimes \mathbf{R}_{B_1} \\ \mathbf{I}_9 - \mathbf{R}_{A_2} \otimes \mathbf{R}_{B_2} \\ \vdots \\ \mathbf{I}_9 - \mathbf{R}_{A_n} \otimes \mathbf{R}_{B_n} \end{pmatrix} vec(\mathbf{R}_X) = \begin{pmatrix} \mathbf{0}_{9\times 1} \\ \mathbf{0}_{9\times 1} \\ \vdots \\ \mathbf{0}_{9\times 1} \end{pmatrix} \tag{1.11}$$

Or, le système (1.10) est de rang plein dès qu'il y a 2 rotations pures d'axes distincts; d'où la proposition suivante.

Théorème 6

Si l'on dispose d'au moins 2 rotations pures d'axes distincts, alors le vecteur de translation $\mathbf{t}_X$ de la transformation pince-caméra est la solution du système de rang plein (1.10).

En ce qui concerne la partie rotationnelle, on a la proposition suivante, dont la preuve se trouve en Annexe C.

Théorème 7

Si l'on dispose d'au moins 2 rotations pures d'axes distincts, alors le système (1.11) est de rang 8, son noyau $\mathcal{K}$ est de dimension 1 et la matrice de rotation $\mathbf{R}_X$ de la transformation pince-caméra est obtenue par :

$$\mathbf{R}_X = \frac{sgn(\det(\mathbf{V}))}{|\det(\mathbf{V})|^{\frac{1}{3}}}\mathbf{V}$$

où sgn est l'opérateur qui renvoie le signe de son argument, $\mathbf{V} = vec^{-1}(\mathbf{v})$ et $\mathbf{v}$ est un vecteur directeur de $\mathcal{K}$.

Remarque Si, au lieu de disposer de rotations pures du robot, on dispose de rotations pures de la caméra, alors on peut se ramener au cas précédent en inversant les rôles de $\mathbf{A}$ et $\mathbf{B}$. Cela nous donnera la transformation inverse de celle que nous cherchons (au lieu de la transformation pince-caméra, nous aurons la transformation caméra-pince).

Conclusion Avec 2 rotations pures d'axes différents, on peut retrouver entièrement la transformation pince-caméra par une solution découplée, ce que fait Li [Li98].

Auto-étalonnage pince-caméra

Cas général

Dans le cas général, ni les rotations, ni les translations ne sont nulles. De (1.6), (1.7) et (1.8), on obtient que la matrice $\mathbf{C}$ du système linéaire (1.6) est égale à la somme de $\mathbf{C}_{trans}$ et de $\mathbf{C}_{rot}$:

$$\mathbf{C} = \mathbf{C}_{trans} + \mathbf{C}_{rot}$$

On a donc l'inégalité :

$$\ker(\mathbf{C}) \leq \ker(\mathbf{C}_{trans}) + \ker(\mathbf{C}_{rot}) \leq 11$$

Un seul mouvement général ne suffit donc pas pour résoudre le problème. Cela est cohérent avec [TL89].

En fait, le rang de $\mathbf{C}$ est 9 (hors cas d'une rotation de π). En effet, on se souviendra que le rang du bloc supérieur gauche de $\mathbf{C}$ ($\mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B$) est 6. Quant au rang de la partie inférieure de $\mathbf{C}$, de dimension $(3 \otimes 12)$, il est égal à 3, car dans le cas présent d'une translation $\mathbf{t}_B$ non nulle, le bloc inférieur gauche est de rang plein et d'après ses dimensions, de rang 3.

Cas de n déplacements généraux Dans le cas de $n > 1$ déplacements généraux, nous formons le système suivant :

$$\begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_{A_1} \otimes \mathbf{R}_{B_1} & \mathbf{0}_{9\times 3} \\ \mathbf{I}_3 \otimes \mathbf{t}_{B_1}^T & \mathbf{I}_3 - \mathbf{R}_{A_1} \\ \mathbf{I}_9 - \mathbf{R}_{A_2} \otimes \mathbf{R}_{B_2} & \mathbf{0}_{9\times 3} \\ \mathbf{I}_3 \otimes \mathbf{t}_{B_2}^T & \mathbf{I}_3 - \mathbf{R}_{A_2} \\ \vdots & \vdots \\ \mathbf{I}_9 - \mathbf{R}_{A_n} \otimes \mathbf{R}_{B_n} & \mathbf{0}_{9\times 3} \\ \mathbf{I}_3 \otimes \mathbf{t}_{B_n}^T & \mathbf{I}_3 - \mathbf{R}_{A_n} \end{pmatrix} vec(\mathbf{X}) = \begin{pmatrix} \mathbf{0}_{9\times 1} \\ \mathbf{t}_{A_1} \\ \mathbf{0}_{9\times 1} \\ \mathbf{t}_{A_2} \\ \vdots \\ \mathbf{0}_{9\times 1} \\ \mathbf{t}_{A_n} \end{pmatrix} \tag{1.12}$$

La partie droite de la matrice du système est de rang 3 dès que l'on dispose de 2 rotations d'axes distincts. La partie gauche est de rang 9 dès que l'on dispose de 2 rotations d'axes distincts et d'une translation.

Donc, avec un minimum d'une rotation pure et d'un mouvement général, le système est de rang plein et a une solution unique.

Cas d'un mouvement général et de n translations pures On voit immédiatement que si l'on ne dispose que d'une seule rotation, le système (1.12) n'est pas de rang plein car la partie droite de sa matrice est de rang 2. Il n'y a donc pas de solution unique. Cependant, nous allons voir que la partie rotationnelle peut être

calculée et que la partie translationnelle appartient à une variété affine de dimension 1.

En supposant que le premier mouvement est un mouvement général et que le second est une translation pure, on peut réécrire le système (1.12) :

$$\begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_{A_1} \otimes \mathbf{R}_{B_1} & \mathbf{0}_{9\times 3} \\ \mathbf{I}_3 \otimes \mathbf{t}_{B_1}^T & \mathbf{I}_3 - \mathbf{R}_{A_1} \\ \mathbf{0}_9 & \mathbf{0}_{9\times 3} \\ \mathbf{I}_3 \otimes \mathbf{t}_{B_2}^T & \mathbf{0}_3 \end{pmatrix} vec(\mathbf{X}) = \begin{pmatrix} \mathbf{0}_{9\times 1} \\ \mathbf{t}_{A_1} \\ \mathbf{0}_{9\times 1} \\ \mathbf{t}_{A_2} \end{pmatrix}$$

qui est équivalent à :

$$\begin{cases} \begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_{A_1} \otimes \mathbf{R}_{B_1} \\ \mathbf{I}_3 \otimes \mathbf{t}_{B_2}^T \end{pmatrix} vec(\mathbf{R}_X) = \begin{pmatrix} \mathbf{0}_{9\times 1} \\ \mathbf{t}_{A_2} \end{pmatrix} \\ (\mathbf{I}_3 - \mathbf{R}_{A_1})\mathbf{t}_X = \mathbf{t}_{A_1} - \mathbf{R}_X \mathbf{t}_{B_1} \end{cases}$$

Si les deux mouvements sont indépendants, (c'est-à-dire que $\mathbf{t}_{B_2}$, l'axe $\mathbf{n}_{B_1}$ de $\mathbf{R}_{B_1}$ et l'axe $\mathbf{n}_{A_1}$ de $\mathbf{R}_{A_1}$ sont indépendants), alors la première équation admet une solution unique pour $\mathbf{R}_X$. En revanche, la deuxième équation admet une infinité de solutions. En effet, c'est un système à 3 inconnues et 3 équations de rang 2. On peut donc ajouter à toute solution un vecteur pris dans le noyau (c.-à-d. proportionnel à $\mathbf{n}_{A_1}$) et obtenir une nouvelle solution. L'ensemble des solutions est donc la variété affine de dimension 1 définie par :

$$\{\mathbf{t}_\perp + \alpha \mathbf{n}_{A_1} | \alpha \in \mathbb{R}\}$$

où $\mathbf{t}_\perp$ est une solution du système perpendiculaire à $\mathbf{n}_{A_1}$. Elle est unique car alors nous résolvons un système de rang 2 dans un espace de dimension 2 (le plan perpendiculaire à $\mathbf{n}_{A_1}$ donc invariant par $\mathbf{R}_{A_1}$). En pratique, cette solution peut être déterminée par une Décomposition en Valeurs Singulières [PTVF92, §2.6] de $\mathbf{I}_3 - \mathbf{R}_{A_1}$.

Cet ensemble de solutions signifie que nous ne pouvons retrouver que la projection de la translation pince-caméra sur le plan invariant associé à l'unique rotation de la caméra. Par exemple, si l'on monte une caméra sur un véhicule routier automatisé, on pourra connaître la position de celle-là par rapport à celui-ci à un unique paramètre près, la hauteur, qui n'est en général pas critique pour ce genre d'application.

On vérifie aisément que ce résultat reste valable quand on remplace le mouvement général par une rotation pure et/ou qu'il y a plus d'une translation.

1.2.3 Interprétation

Les résultats théoriques précédents sont rassemblés dans le Tableau 1.1. Il en ressort que tout type de mouvement contribue à la détermination de l'orientation

relative de la pince par rapport à la caméra alors que seules les rotations participent à la détermination de sa position relative. Ceci explique que cette dernière est en général difficile à obtenir précisément.

Une autre remarque intéressante est que cette étude algébrique confirme l'analyse géométrique du problème présenté dans [Che91b]. De plus, on trouvera dans [Wan92] une exploitation de certains des cas exhibés par notre étude.

	Translation $\mathbf{R}_B = \mathbf{I}$ $\mathbf{t}_B \neq 0$	Rotation $\mathbf{R}_B \neq \mathbf{I}$ $\mathbf{t}_B = 0$	Mouvement général $\mathbf{R}_B \neq \mathbf{I}$ $\mathbf{t}_B \neq 0$
Translation $\mathbf{R}_B = \mathbf{I}$ $\mathbf{t}_B \neq 0$	$\mathbf{R}_X$ (si l'on dispose d'une troisième translation indépendante)	$\mathbf{R}_X$ $\mathbf{t}_X(\alpha)$	$\mathbf{R}_X$ $\mathbf{t}_X(\alpha)$
Rotation $\mathbf{R}_B \neq \mathbf{I}$ $\mathbf{t}_B = 0$	$\mathbf{R}_X$ $\mathbf{t}_X(\alpha)$	$\mathbf{R}_X, \mathbf{t}_X$ Solution découplée	$\mathbf{R}_X, \mathbf{t}_X$ Solution du cas général
Mouvement général $\mathbf{R}_B \neq \mathbf{I}$ $\mathbf{t}_B \neq 0$	$\mathbf{R}_X$ $\mathbf{t}_X(\alpha)$	$\mathbf{R}_X, \mathbf{t}_X$ Solution du cas général	$\mathbf{R}_X, \mathbf{t}_X$ Solution du cas général

TAB. 1.1: *Résumé des résultats pour 2 mouvements indépendants*

1.2.4 Résolution effective

Séparons le système (1.12) en

$$\begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_{A_1} \otimes \mathbf{R}_{B_1} \\ \mathbf{I}_9 - \mathbf{R}_{A_2} \otimes \mathbf{R}_{B_2} \\ \vdots \\ \mathbf{I}_9 - \mathbf{R}_{A_n} \otimes \mathbf{R}_{B_n} \end{pmatrix} vec(\mathbf{R}_X) = 0 \tag{1.13}$$

et

$$\begin{pmatrix} \mathbf{I}_3 - \mathbf{R}_{A_1} \\ \mathbf{I}_3 - \mathbf{R}_{A_2} \\ \vdots \\ \mathbf{I}_3 - \mathbf{R}_{A_n} \end{pmatrix} \mathbf{t}_X = \begin{pmatrix} \mathbf{t}_{A_1} - \mathbf{R}_X \mathbf{t}_{B_1} \\ \mathbf{t}_{A_2} - \mathbf{R}_X \mathbf{t}_{B_2} \\ \vdots \\ \mathbf{t}_{A_n} - \mathbf{R}_X \mathbf{t}_{B_n} \end{pmatrix} \tag{1.14}$$

On peut alors résoudre le problème en 2 phases : on retrouve d'abord la rotation selon la Proposition 7, puis la translation connaissant la rotation.

Cette solution possède plusieurs avantages. Le premier est un avantage par rapport à une résolution obtenue par moindres carrés linéaires du système (1.12). En effet, en présence de bruit, la solution par moindres carrés linéaires va nous donner un vecteur de dimension 12, dont les 9 coefficients associés à la rotation ne satisferont pas les contraintes d'orthogonalité (cf. Annexe C). On pourra certes les appliquer *a posteriori* mais, comme ces contraintes ne sont pas linéaires, nous ne saurons pas quelle correction apporter à la translation. En revanche, la solution en 2 pas garantit l'obtention d'une rotation "propre". De plus, cette rotation est obtenue par l'intermédiaire d'un vecteur directeur d'un noyau de dimension 1 d'une application linéaire qui se calcule très robustement par Décomposition en Valeurs Singulières. Ainsi donc, la translation sera obtenue par moindres carrés linéaires sur un espace linéaire de dimension 3, qui est optimale.

Un autre avantage de la résolution en 2 étapes est qu'elle tient compte du cas particulier où le robot n'effectue que des rotations pures. En effet, elle devient alors équivalente à la solution découplée que nous avons vu dans ce cas particulier.

Enfin, elle permet de détecter le cas particuliers où le robot n'effectue que des translations pures car alors l'équation matricielle (1.13) devient : $\mathbf{0}\ vec(\mathbf{R}_X) = 0$.

1.3 Mesure d'erreur

L'espace des déplacements euclidiens $SE(3)$ ne possédant pas de métrique naturelle, il nous a fallu faire un choix. Nous avons donc défini séparément une erreur relative en rotation et une erreur relative en translation.

erreur	10^{-5}	10^{-4}	10^{-3}	10^{-2}
angle (deg)	0.3624	1.1459	3.6239	11.4639
angle (deg)	5	10	15	20
erreur	1.9036 10^{-3}	7.6106 10−3	1.7110 10−2	3.0384 10−2

TAB. 1.2: *Quelques couples (erreur en rotation, angle résiduel).*

L'erreur relative en rotation a été calculée par : $\|\hat{\mathbf{q}} - \mathbf{q}\|$ où $\mathbf{q}$ est le quaternion associé à la rotation pince-caméra obtenue par étalonnage selon la méthode utilisée et $\hat{\mathbf{q}}$ est le quaternion associé à la rotation pince-caméra de référence. On notera que si les quaternions $\hat{\mathbf{q}}$ et $\mathbf{q}$ sont unitaires, alors

$$\begin{aligned}
\|\hat{\mathbf{q}} - \mathbf{q}\| &= \|\mathbf{1} - \hat{\mathbf{q}}^{-1}\mathbf{q}\| \\
&= (\mathbf{1} - \hat{\mathbf{q}}^{-1}\mathbf{q})\overline{(\mathbf{1} - \hat{\mathbf{q}}^{-1}\mathbf{q})} \\
&= (\mathbf{1} - \hat{\mathbf{q}}^{-1}\mathbf{q})(\mathbf{1} - \overline{\hat{\mathbf{q}}^{-1}\mathbf{q}}) \\
&= \mathbf{1} + (\hat{\mathbf{q}}^{-1}\mathbf{q})(\overline{\hat{\mathbf{q}}^{-1}\mathbf{q}}) \\
&\quad -(\hat{\mathbf{q}}^{-1}\mathbf{q} + \overline{\hat{\mathbf{q}}^{-1}\mathbf{q}}) \\
&= 2 - 2\cos(\frac{\alpha}{2})
\end{aligned}$$

où α est l'angle de la rotation résiduelle $\hat{\mathbf{q}}^{-1}\mathbf{q}$. Cette fonction est strictement croissante et vaut 0 au voisinage de $\alpha = 0$. De plus, elle permet d'éviter l'indétermination apparaîssant pour $\alpha = \pi$ lors de l'utilisation d'une distance géodésique (voir [SLBE91], p.35). Enfin, l'étude de sa fonction réciproque ($\epsilon \mapsto 2\arccos(1 - \frac{\epsilon}{2}) = 2\sqrt{\epsilon} + \mathcal{O}(\epsilon^{3/2})$) permet de rendre un peu plus tangible le résultat d'une telle mesure d'erreur (voir Tableau 1.2).

L'erreur relative en translation a été obtenue par: $\|\mathbf{t} - \mathbf{t}^*\|/\|\mathbf{t}^*\|$ où $\mathbf{t}$ et $\mathbf{t}^*$ représentent respectivement les translations pince-caméra estimée et réelle.

Les erreurs finales ont été obtenues en faisant la moyenne des carrés des erreurs relatives (RMS).

1.4 Simulations

Pour chaque série de simulations, la démarche a été similaire : pour chaque valeur du paramètre testé, nous avons tiré aléatoirement une transformation pince-caméra (par tirage aléatoire des angles Roulis-Tangage-Lacet de la matrice de rotation et des coefficients de translation selon des lois normales), une séquence de mouvements conjugués du robot et de la caméra, puis nous avons bruité les déplacements de la caméra, effectué l'étalonnage avec 3 méthodes différentes et comparé leur résultat avec la transformation initiale.

Les méthodes utilisées sont la méthode axe/angle [TL89], la méthode des quaternions duaux unitaires [DBC96], la méthode d'optimisation non-linéaire [HD95] et la méthode linéaire présentée dans ce rapport.

1.4.1 Influence du bruit

Description Dans cette simulation, nous avons bruité les mouvements de la caméra $(\mathbf{R}_{A_i}, \mathbf{t}_{A_i})$ et obtenu les mouvements bruités $(\tilde{\mathbf{R}_{A_i}}, \tilde{\mathbf{t}_{A_i}})$.

Les translations $\tilde{\mathbf{t}_{A_i}}$ ont été déterminées par :

$$\tilde{\mathbf{t}_{A_i}} = \mathbf{t}_{A_i} + bruit * \|\mathbf{t}_{A_i}\| * \mathbf{n}$$

où *bruit* est un facteur variant de 0 à 0.1 et $\mathbf{n}$ est une réalisation d'un vecteur aléatoire de loi normale centrée réduite.

Quant aux rotations, nous les avons bruitées en ajoutant une erreur relative sur leurs angles Roulis-Tangage-Lacet :

$$\tilde{\alpha} = \alpha(1 + bruit * r)$$

où α est un angle de la rotation exacte, $\tilde{\alpha}$ l'angle correspondant de la rotation bruitée, *bruit* est le même facteur que pour la translation. Enfin, r est une réalisation d'une variable aléatoire de loi normale centrée réduite.

Étalonnage avec 2 mouvements quelconques Nous avons effectué 100 choix différents de transformations pince-caméra et de couples de mouvements. Pour chaque choix, nous avons bruité les mouvements de la caméra avec des *bruit* variant de 0 à 0.2. La Figure 1.2 présente les erreurs d'étalonnage obtenues selon les 4 méthodes retenues avec 2 mouvements et la figure 1.3 les temps de calculs moyens respectifs.

Avec seulement 2 mouvements, les méthodes linéaires sont plus précises en rotation mais moins robustes en translation que les méthodes plus complexes.

Étalonnage avec 2 mouvements de faible amplitude Cette simulation est en tous points semblable avec la précédente, exception faite de l'amplitude des mouvements d'étalonnage que nous avons limitée en rotation et translation. Les erreurs d'étalonnage sont présentées en Figure 1.4.

Les résultats obtenus sont sensiblement meilleurs que pour les méthodes complexes (quaternions duaux et minimisation non-linéaire) et légèrement meilleurs que pour l'autre méthode linéaire : méthode axe/angle.

1.4.2 Influence du nombre de mouvements

Description Nous avons ici fait varier le nombre de mouvements de 2 à 15 en gardant un niveau de bruit constant ($bruit = 0.01$, selon la définition du paragraphe

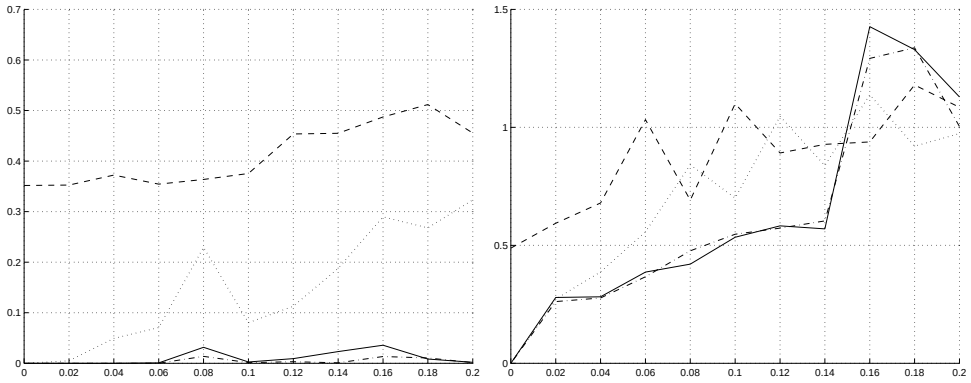


FIG. 1.2: *Erreurs en rotation (gauche) et en translation (droite) pour l'étalonnage en fonction du niveau de bruit dans le cas de 2 mouvements du robot selon les différentes méthodes : axe/angle (—), quaternions (· · ·), non-linéaire (- -), linéaire (— ·).*

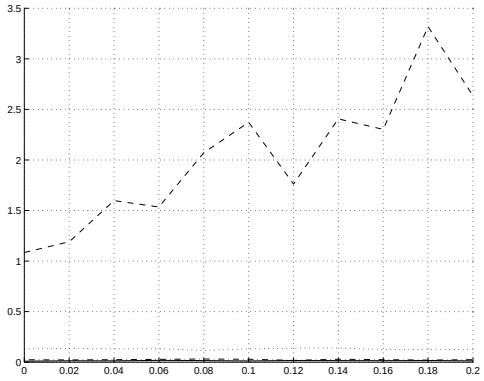


FIG. 1.3: *Temps moyen de calcul (en secondes CPU) associés aux résultats de la Figure 1.2.*

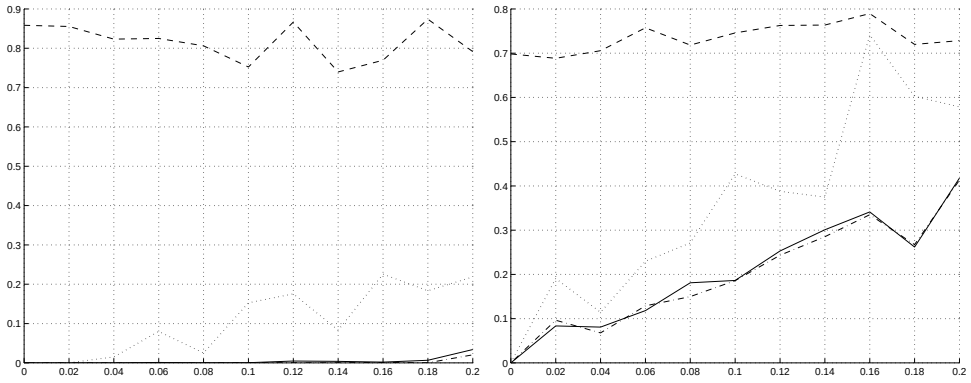


FIG. 1.4: *Erreurs en rotation (gauche) et en translation (droite) pour l'étalonnage en fonction du niveau de bruit dans le cas de 2 mouvements* **de faible amplitude** *du robot selon les différentes méthodes : axe/angle (—), quaternions (· · ·), non-linéaire (- -), linéaire (– ·).*

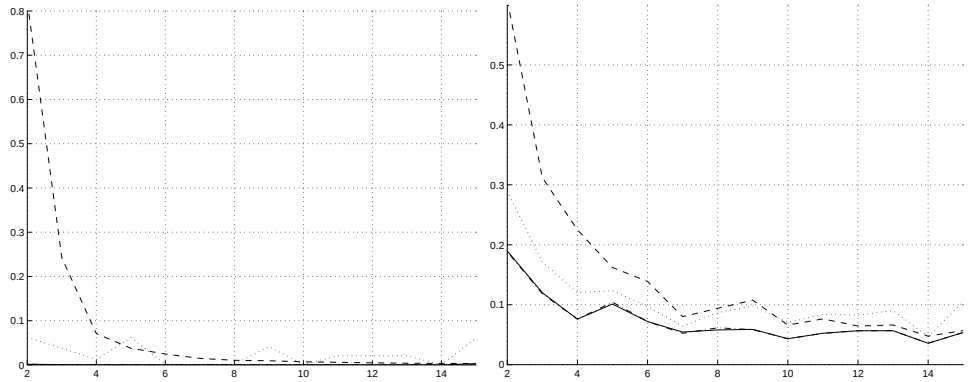


FIG. 1.5: *Erreurs en rotation (gauche) et en translation (droite) pour l'étalonnage en fonction du nombre de mouvements du robot (rotations pures de faible amplitude) selon les différentes méthodes : axe/angle(—), quaternions (· · ·), non-linéaire (- -), linéaire (– ·).*

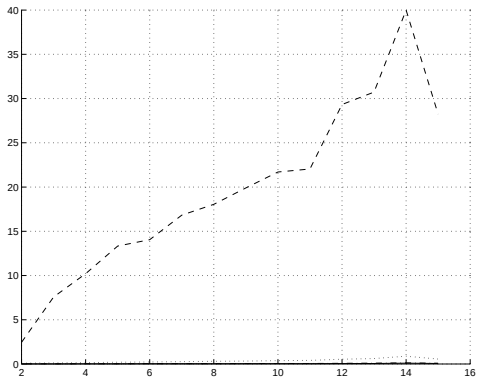


FIG. 1.6: *Temps moyen de calcul (en secondes CPU) pour l'étalonnage en fonction du nombre de mouvements du robot (rotations pures de faible amplitude) selon les différentes méthodes : axe/angle (—), quaternions (· · ·), non-linéaire (- -), linéaire (– ·).*

précédent). Nous nous sommes concentrés ici sur des mouvements de faible amplitude. Les mouvements de caméra ont été bruités de la même manière que pour la simulation précédente et 100 tirages aléatoires ont été fait pour chaque nombre de mouvements.

La figure 1.5 présente les erreurs d'étalonnage obtenues selon les 3 méthodes retenues et la figure 1.6, les temps de calculs moyens respectifs.

Les résultats sont semblables à ce que l'on a vu pour l'influence du bruit : une erreur bien plus faible en rotation, une erreur très faible (et inférieure à celles des autres méthodes) en translation et un temps de calcul constant et réduit.

1.5 Résultats expérimentaux

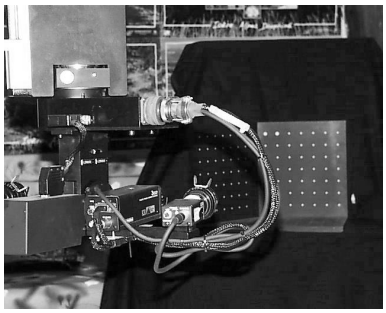


FIG. 1.7: *Dispositif expérimental : une caméra montée sur un robot cartésien à 5 ddl regarde une mire*

Nous avons utilisé un robot cartésien à 5 degrés de liberté portant une caméra (Figure 1.7). À l'aide d'une connaissance *a priori* mais néanmoins imprécise de la transformation pince-caméra, nous avons déplacé le robot de manière à ce que la caméra conserve la mire dans son champ de vision. En chacune des positions du robot, nous avons enregistré la valeur de ses coordonnées articulaires et pris une image de la mire.

Munis de ces précieuses informations, nous avons fastidieusement procédé à l'étalonnage pince-caméra. Pour cela, nous avons transformé les coordonnées articulaires du robot en positions dans l'espace puis en déplacements. Puis, nous avons détecté les points de la mire dans l'image et les avons mis en correspondance avec le modèle géométrique que nous en possédons. Nous avons alors déterminé les paramètres de formation de l'image, parmi lesquels on trouve la pose de la caméra par rapport à la mire. De ces poses, nous avons enfin pu déterminer les déplacements de la caméra. Ce fut alors un jeu d'enfant de déterminer la transformation pince-caméra.

Afin de mesurer l'erreur d'étalonnage et comme nous ne possédons pas de valeur de référence, nous avons calculé, pour chaque couple de déplacement $(\mathbf{A}, \mathbf{B})$, l'écart entre les transformations rigides $\mathbf{AX}$ et $\mathbf{XB}$ selon les mesures définies au paragraphe 1.4.

1.5.1 Trajectoire 1

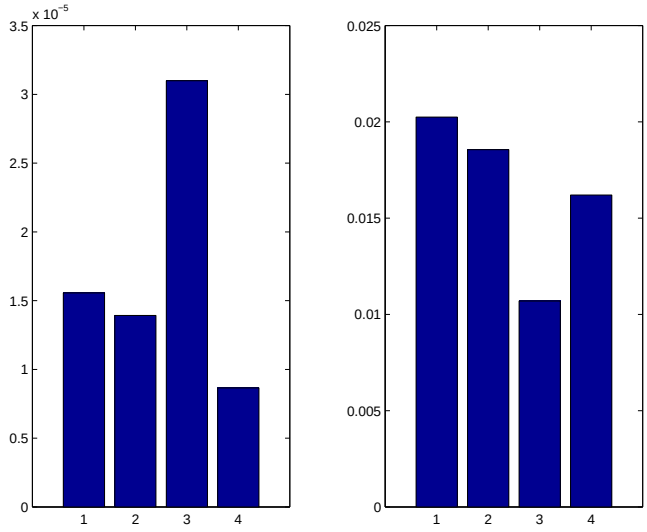


FIG. 1.8: *Moyenne des erreurs en rotation (gauche) et en translation (droite) pour une trajectoire de 33 positions. Méthodes utilisées :* **1)** *axe/angle [TL89],* **2)** *quaternions duaux [DBC96],* **3)** *non linéaire [HD95],* **4)** *linéaire.*

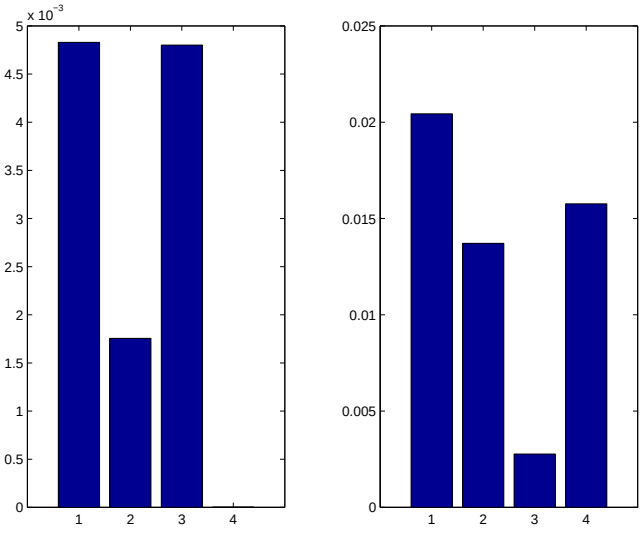


FIG. 1.9: *Moyenne des erreurs en rotation (gauche) et en translation (droite) pour une trajectoire composée de 2 grands mouvements puis 60 petits et exponentiellement décroissants. Mêmes conventions qu'en Figure 1.8.* **NB:** *l'erreur en rotation selon* **4)** *n'a pas été oubliée !*

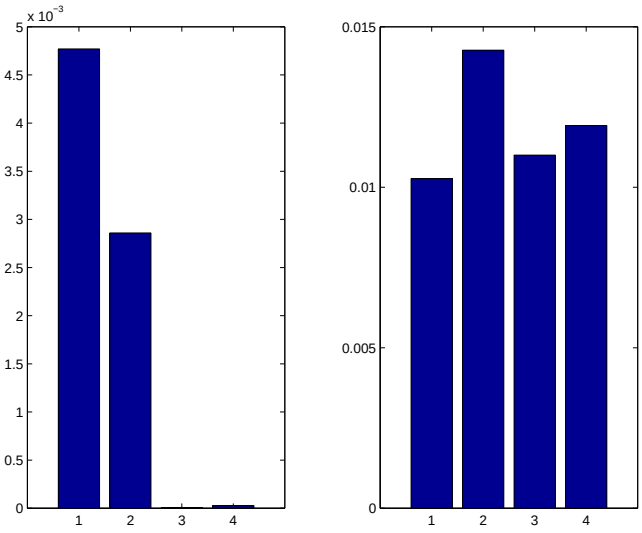


FIG. 1.10: *Moyenne des erreurs en rotation (gauche) et en translation (droite) pour une trajectoire de 60 mouvements, petits et exponentiellement décroissants. Mêmes conventions qu'en Figure 1.8.*

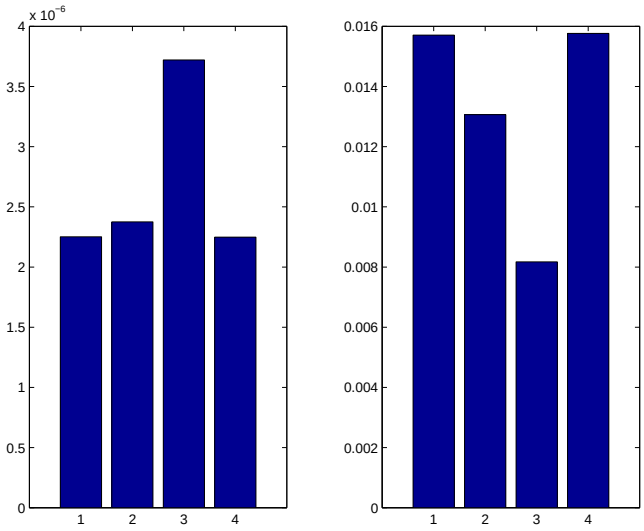


FIG. 1.11: *Moyenne des erreurs en rotation (gauche) et en translation (droite) avec 2 mouvements de grande amplitude. Mêmes conventions qu'en Figure 1.8.* **NB:** *L'erreur en rotation de* **3)** *a été divisée par 1000 pour des raisons graphiques.*

Dans cette expérience, nous avons utilisé une trajectoire de 33 positions les plus éloignées possibles (au sens de [TL89]). La Figure 1.8 montre la moyenne des erreurs en rotation (gauche) et en translation (droite) obtenues en étalonnant selon **1)** la méthode axe/angle [TL89], **2)** la méthode des quaternions duaux [DBC96], **3)** la minimisation non linéaire [HD95] et **4)** la méthode linéaire présentée ici.

Nous avons utilisé cette séquence d'images pour calculer de façon robuste (§ C.2) une transformation pince-caméra qui nous servira d'étalon pour les expériences suivantes.

1.5.2 Trajectoire 2

Dans cette deuxième expérience, nous avons utilisé une trajectoire composée de 2 mouvements initiaux de grande amplitude et d'axes de rotation différents puis de 60 mouvements de faible amplitude convergeant exponentiellement vers 0 selon les coordonnées articulaires du robot. La Figure 1.9 montre les erreurs en rotation et translation (mêmes conventions qu'en Figure 1.8). Quant à la Figure 1.10, elle montre les erreurs lorsque l'on n'utilise que les mouvements de faible amplitude (trajectoire partielle **expo**). Enfin, la Figure 1.11 montre les erreurs lorsque l'on n'utilise que les 2 mouvements initiaux (trajectoire partielle **init**). Les erreurs par rapport à la transformation pince-caméra moyenne obtenue avec les images de la séquence précédente ont été collectées dans le Tableau 1.3.

Les deux types d'erreurs donnent lieu à une conclusion identique si l'on compare

	Figure 1.9 : complète		Figure 1.10 : **expo**		Figure 1.11 : **init**	
	rot.	trans.	rot.	trans.	rot.	trans.
1	1.75	1.06	1.75	1.01	$1.48\ 10^{-5}$	0.56
2	1.60	1.01	1.42	1.00	$4.94\ 10^{-5}$	0.41
3	1.81	1.00	$4.91\ 10^{-4}$	1.01	1.39	1.01
4	$1.00.10^{-5}$	0.57	$6.23\ 10^{-3}$	0.87	$1.04\ 10^{-5}$	0.56

TAB. 1.3: *Erreurs par rapport à la transformation moyenne*

les deux trajectoires partielles : la méthode non-linéaire n'est pas efficace avec peu de mouvements (même grands) mais l'est avec de nombreux mouvements (mêmes petits); la méthode linéaire que nous proposons est efficace dans les deux cas et les deux autres méthodes perdent leur efficacité lorsque les mouvements sont petits.

Enfin, ce manque de précision sur une partie de la trajectoire pour les méthodes classiques se répercute sur les résultats obtenus avec la trajectoire complète.

1.6 Conclusion

Dans ce chapitre, nous avons donné une formulation purement linéaire du problème de l'étalonnage pince-caméra. Cette formulation nous a permis une analyse algébrique simple du problème. Nous avons ainsi pu dresser un tableau de ce que l'on peut obtenir en fonction du type de mouvement effectués par le robot.

Les résultats de simulation et expérimentaux montrent que notre méthode linéaire donne de meilleurs résultats en rotation que les méthodes de référence. En ce qui concerne la translation, ce n'est pas pire avec notre méthode qu'avec les autres.

Enfin, cette formulation linéaire nous sert de base pour définir une méthode d'auto-étalonnage (Chapitre 2) et une méthode d'étalonnage en ligne (Chapitre 3) de la transformation pince-caméra.

Chapitre 2

Auto-étalonnage pince-caméra

2.1 Introduction

Les méthodes d'étalonnage classiques, y compris celle présentée au chapitre précédent et à l'exception de [WAH98] qui utilise une machinerie mathématique complexe, ne fonctionnent qu'à condition que la caméra observe une scène dont la structure est connue. Cela se traduit par l'utilisation d'une mire de calibration.

Notre but dans ce chapitre est de présenter des moyens, que nous regrouperons sous le terme d' « auto-étalonnage pince-caméra », de se passer de cette mire de calibration. En cela, cette méthode autorise une plus grande autonomie du robot.

Pour que cette autonomie soit totale, il faut pouvoir étalonner la transformation pince-caméra sans connaissance *a priori* sur sa valeur. Dans ce cadre, on ne peut donc pas effectuer de grands mouvements car on risquerait de n'avoir jamais la même scène dans le champ de la caméra, ce qui empêcherait tout calcul de reconstruction tridimensionnelle. Il faut donc que la méthode soit robuste aux petits mouvements. Nous allons donc modifier la méthode linéaire proposée au chapitre précédent qui possède cette propriété.

Nous avons identifié jusqu'à présent deux types d'auto-étalonnage pince-caméra. Le premier est le plus général qui soit puisqu'il s'agit de déplacer la caméra devant une scène totalement inconnue. Dans ce cas, on ne peut calculer le déplacement de la caméra qu'à un facteur d'échelle près.

Le second est plus spécifique, mais d'un grand intérêt au regard de la première partie de cette thèse, car il permet la détermination de la transformation pince-caméra par l'observation d'un faisceau de droites. Ici, le déplacement de la caméra dépend de deux facteurs d'échelle inconnus.

2.2 Scène inconnue

Dans ce paragraphe, nous allons modifier la méthode linéaire du chapitre précédent de manière à pouvoir effectuer l'étalonnage pince-caméra face à une scène inconnue, c.-à-d. en se passant d'une mire de calibration. Ceci simplifie la procédure pratique d'étalonnage, présentée en Figure 2.1. Désormais, à partir d'images d'une scène inconnue, on effectue une reconstruction euclidienne. Cela fournit les positions relatives, à un facteur d'échelle près, de la caméra par rapport à sa position initiale. De celles-ci, on ne peut tirer les déplacements de la caméra qu'à ce facteur d'échelle inconnu, mais pourtant retrouver la transformation pince-caméra.

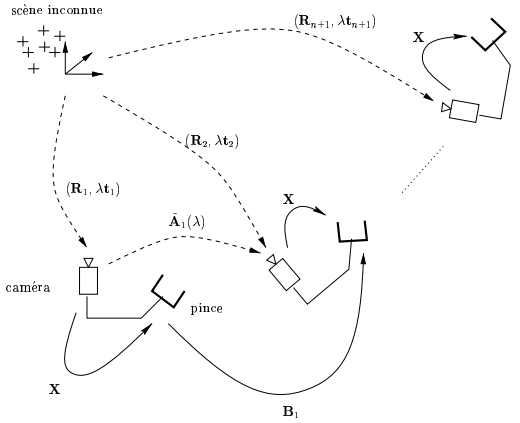


FIG. 2.1: *La procédure d'étalonnage pince-caméra dans le cas d'une scène inconnue (à comparer avec celle présentée en Figure 1.1).*

Les contraintes de cette nouvelle méthode se restreignent donc à celles de la reconstruction euclidienne :

- la connaissance des paramètres intrinsèques de la caméra avec une précision pas nécessairement très grande ;
- la capacité à suivre un ensemble de points rigidement liés au cours du déplacement de la caméra.

Ainsi, face à une scène inconnue, nous pouvons calculer les déplacements de caméra par le biais d'une reconstruction tridimensionnelle, mais celle-ci ne peut se faire qu'à un facteur d'échelle près. Dans ces conditions, l'équation (1.3) devient :

$$(\mathbf{I}_3 - \mathbf{R}_A)\mathbf{t}_X = \lambda\widetilde{\mathbf{t}_A} - \mathbf{R}_X\mathbf{t}_B \tag{2.1}$$

où λ est le facteur d'échelle inconnu, $\widetilde{\mathbf{t}_A}$ est la translation reconstruite de la caméra et $\mathbf{t}_A = \lambda\widetilde{\mathbf{t}_A}$ est la translation réelle effectuée par la caméra.

Il est bien évident que l'utilisation d'une méthode classique d'étalonnage pince-caméra donnerait lieu à une solution biaisée pour la translation pince-caméra $\mathbf{t}_X$. En effet, l'équation précédente se réécrit :

$$\underbrace{(\mathbf{I}_3 - \mathbf{R}_A)\mathbf{t}_X = \widetilde{\mathbf{t}_A} - \mathbf{R}_X\mathbf{t}_B}_{\text{partie classique}} + \underbrace{(\lambda - 1)\widetilde{\mathbf{t}_A}}_{biais}$$

Il est donc nécessaire de modifier les méthodes existantes afin de tenir compte de ce facteur d'échelle inconnu.

Ainsi, notre méthode linéaire en deux étapes peut être légèrement modifiée afin de pouvoir établir simultanément la transformation pince-caméra et le facteur d'échelle inconnu de la reconstruction.

2.2.1 Nouvelle formulation

Repartons du système (1.6) en remplaçant $\mathbf{t}_A$ par $\lambda\widetilde{\mathbf{t}_A}$ pour obtenir :

$$\begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B & \mathbf{0}_{9\times 3} \\ \mathbf{I}_3 \otimes (\mathbf{t}_B^T) & \mathbf{I}_3 - \mathbf{R}_A \end{pmatrix} \begin{pmatrix} vec(\mathbf{R}_X) \\ \mathbf{t}_X \end{pmatrix} = \begin{pmatrix} \mathbf{0}_{9\times 1} \\ \lambda\widetilde{\mathbf{t}_A} \end{pmatrix}$$

Comme le second membre dépend du facteur inconnu λ, nous insérons ce dernier dans le vecteur des inconnues pour obtenir :

$$\begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_A \otimes \mathbf{R}_B & \mathbf{0}_{9\times 3} & \mathbf{0}_{3\times 1} \\ \mathbf{I}_3 \otimes \mathbf{t}_B^T & \mathbf{I}_3 - \mathbf{R}_A & -\widetilde{\mathbf{t}_A} \end{pmatrix} \begin{pmatrix} vec(\mathbf{R}_X) \\ \mathbf{t}_X \\ \lambda \end{pmatrix} = \mathbf{0}_{12\times 1} \tag{2.2}$$

Nous pouvons donc encore résoudre ce système en 2 étapes : d'abord, la rotation (étape inchangée) puis la translation.

Une question se pose lorsque nous considérons plusieurs mouvements : doit-on déterminer un ou plusieurs facteurs d'échelle inconnus ? En pratique, les méthodes de reconstruction ne fournissent qu'un seul facteur d'échelle par séquence d'images utilisée.

Nous ne nous intéressons ici qu'à retrouver la translation et le facteur d'échelle connaissant la rotation. Nous devons donc résoudre :

$$\begin{pmatrix} \mathbf{I}_3 - \mathbf{R}_{A_1} & -\widetilde{\mathbf{t}_{A_1}} \\ \vdots & \\ \mathbf{I}_3 - \mathbf{R}_{A_n} & -\widetilde{\mathbf{t}_{A_n}} \end{pmatrix} \begin{pmatrix} \mathbf{t}_X \\ \lambda \end{pmatrix} = \begin{pmatrix} -\mathbf{R}_X\mathbf{t}_{B_1} \\ \vdots \\ -\mathbf{R}_X\mathbf{t}_{B_n} \end{pmatrix} \tag{2.3}$$

Ce système est de rang plein avec au moins 2 rotations indépendantes et une translation et peut alors être résolu par moindres carrés linéaires sur un espace linéaire désormais de dimension 4. Cette condition étant compatible avec les conditions d'existence d'une solution pour la rotation, nous pouvons donc effectuer pleinement l'étalonnage pince-caméra et déterminer la taille de la scène observée.

Auto-étalonnage pince-caméra

2.2.2 Cas des rotations pures du robot

Si le robot n'effectue que des rotations pures, alors le second membre de l'équation (2.3) s'annule. Quelles en sont les conséquences? Revenons à l'équation (2.1) et remarquons qu'elle devient alors :

$$(\mathbf{I}_3 - \mathbf{R}_A)\mathbf{t}_X - \lambda\widetilde{\mathbf{t}_A} = 0$$

ce qui signifie que les coefficients de $\mathbf{t}_X$ et λ sont liés. Le système (2.3) avec un second membre nul n'est donc pas de rang plein.

Interprétation Contrairement au cas où l'on utilise une mire, il n'est plus possible ici d'étalonner entièrement la relation pince-caméra uniquement avec des rotations pures du robot. Cela s'explique par le fait que les seules informations métriques dont nous puissions disposer lorsque ni la scène, ni la transformation pince-caméra ne sont connues proviennent exclusivement des déplacements du robot. Or, parmi ceux-ci, les rotations pures sont dépourvues de la notion de longueur. Il faut donc avoir recours aux translations pour pouvoir obtenir le facteur d'échelle inconnu.

2.2.3 Résultats expérimentaux

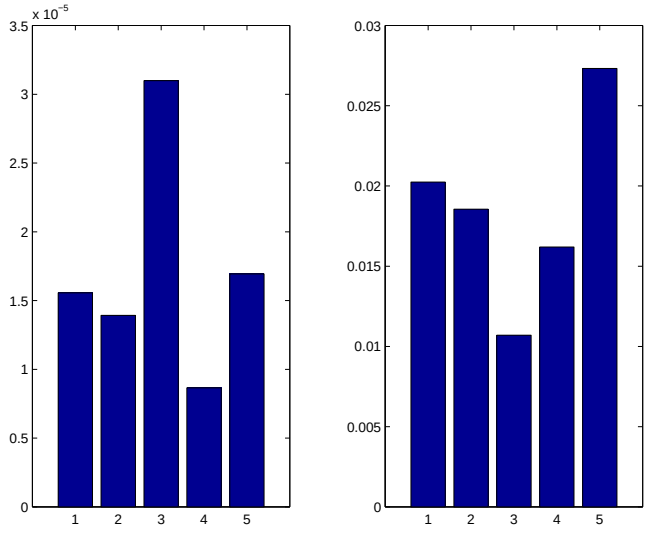


FIG. 2.2: *Moyenne des erreurs en rotation (gauche) et en translation (droite) pour une trajectoire de 33 positions. Méthodes utilisées :* **1)** *axe/angle [TL89],* **2)** *quaternions duaux [DBC96],* **3)** *non linéaire [HD95],* **4)** *linéaire,* **5)** *auto-étalonnage.*

Afin de comparer les résultats de la méthode d'auto-étalonnage pince-caméra avec la méthode linéaire présentée dans le chapitre précédent, nous les avons appliquées, dans un premier temps, aux mêmes images (de la mire, donc). La différence

d'utilisation de ces images réside dans l'utilisation ou non du modèle géométrique de la mire pour le calcul des déplacements de la caméra. En revanche, nous avons conservé, pour la méthode d'auto-étalonnage, les points issus de la détection des cibles de la mire (utilisant le modèle de la mire) utilisée dans l'étalonnage classique.

La Figure 2.2 reprend la Figure 1.8 et y ajoute le résultat obtenu par la méthode d'auto-étalonnage. La méthode d'auto-étalonnage est moins précise du fait de la non utilisation du modèle géométrique de la mire. Cependant, les résultats qu'elle fournit restent honorables puisque l'erreur relative à la transformation moyenne calculée est de 2.10^{-4} en rotation (correspondant à un angle résiduel d'environ un demi degré) et de 32% en translation.

Dans un second temps, nous nous sommes affranchis complètement de la mire et avons travaillé sur des images réelles. Pour cela, nous avons fait suivre au robot une trajectoire au cours de laquelle nous avons pris des images de la mire, grâce auxquelles nous avons procédé à l'étalonnage par les méthodes classiques et linéaire. Puis, nous avons fait reparcourir cette même trajectoire au robot (à sa répétabilité près) pour prendre des images d'une scène inconnue sur lesquelles nous avons utilisé la méthode d'auto-étalonnage.

Nous avons utilisé deux trajectoires, une de 4 positions éloignées et une autre de 9 positions proches (voir Figures 2.3 et 2.4). On notera ici encore que la méthode linéaire d'étalonnage avec mire est toujours aussi efficace (Figures 2.5 et 2.6 et Tableau 2.1). En ce qui concerne la méthode d'auto-étalonnage, les résultats sont quelque peu moins bons mais restent acceptables comparés aux méthodes avec mire.

	4 positions		9 positions	
	rot.	trans.	rot.	trans
1	$2.7.10^{-5}$	0.18	0.62	0.99
2	$2.8.10^{-5}$	0.22	$4.7.10^{-4}$	0.41
3	1.82	1.01	0.02	0.44
4	$2.3.10^{-5}$	0.17	$2.3.10^{-3}$	0.54
5	$2.8.10^{-4}$	0.20	0.02	0.97

TAB. 2.1: *Erreurs par rapport à la transformation moyenne.*

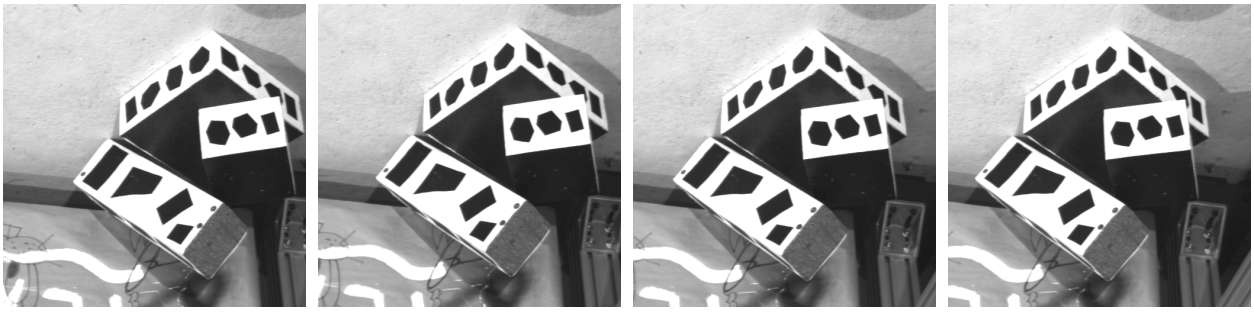


FIG. 2.3: *La séquence de 4 images utilisées pour l'auto-étalonnage*

FIG. 2.4: *La séquence de 9 images utilisées pour l'auto-étalonnage*

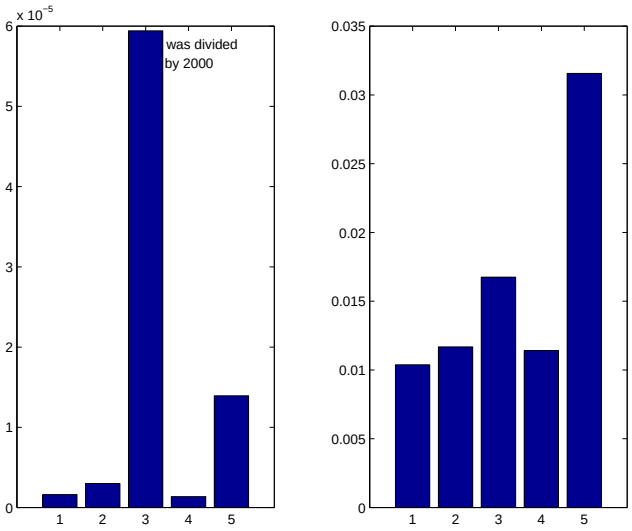


FIG. 2.5: *Moyenne des erreurs en rotation (gauche) et en translation (droite) pour une séquence de 4 positions éloignées. Méthodes utilisées :* **1)** *axe/angle [TL89],* **2)** *quaternions duaux [DBC96],* **3)** *non linéaire [HD95],* **4)** *linéaire,* **5)** *auto-étalonnage.*

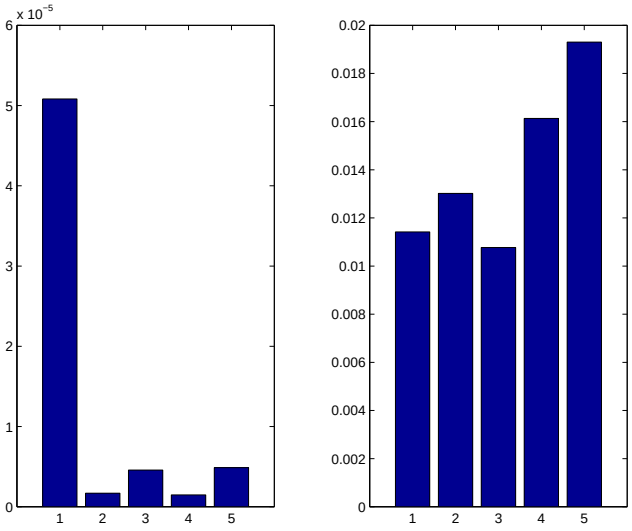


FIG. 2.6: *Moyenne des erreurs en rotation (gauche) et en translation (droite) pour une trajectoire de 9 positions rapprochées. Mêmes conventions que pour la Figure 2.5.*

Auto-étalonnage pince-caméra

2.3 Faisceau de droites

Nous nous plaçons ici dans les mêmes conditions que dans la première partie de cette thèse, à savoir qu'une caméra étalonnée observe un faisceau de droites concourantes. On peut alors, pour chaque image, calculer une pose partielle, c.-à-d. la pose de la caméra sans aucune notion d'échelle. Ceci introduit un facteur d'échelle différent pour chaque pose (voir Figure 2.7). Les mouvements de la caméra dépendent alors de deux facteurs d'échelles inconnus. Pourtant, la transformation pince-caméra reste calculable grâce aux mouvements du robot.

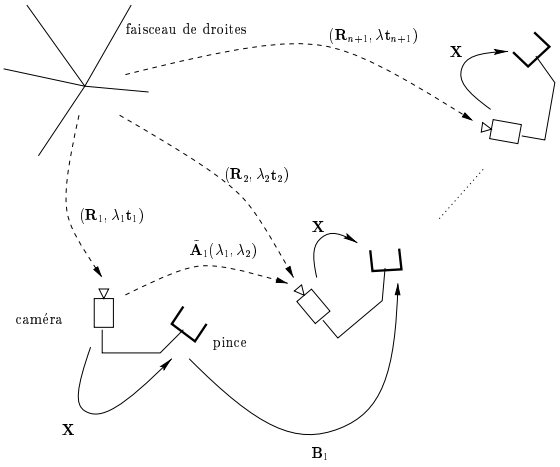


FIG. 2.7: *Étalonnage pince-caméra lorsque la caméra observe un faisceau de droites concourantes. (Comparer aux Figures 1.1 et 2.1)*

2.3.1 Mise en équation

Notons $\mathbf{P}_i(\lambda_i) = (\mathbf{R}_i, \lambda_i \mathbf{t}_i)$, la pose partielle, dépendant du facteur d'échelle inconnu λ_i, obtenue avec l'image i. Alors, le déplacement i de la caméra de la position i à la position $i+1$ est donné par :

$$\mathbf{A}_i(\lambda_i, \lambda_{i+1}) = \begin{pmatrix} \mathbf{R}_{i+1}\mathbf{R}_i^T & \lambda_{i+1}\mathbf{t}_{i+1} - \lambda_i \mathbf{R}_{i+1}\mathbf{R}_i^T \mathbf{t}_i \\ 0 & 1 \end{pmatrix}$$

Les facteurs d'échelle inconnus n'interviennent que dans la translation. Par conséquent, nous pouvons conserver la première étape de calcul de la rotation pince-caméra de la méthode linéaire. En revanche, il nous faut modifier la deuxième étape qui donne la translation pince-caméra.

En gardant les mêmes notations que précédemment, la partie translationnelle (1.3) de l'équation de conjugaison entre mouvement de la caméra et mouvement du robot

s'écrit pour tout i :

$$\mathbf{R}_{ai}\mathbf{t}_x + \lambda_{i+1}\mathbf{t}_{i+1} - \lambda_i\mathbf{R}_{ai}\mathbf{t}_i = \mathbf{R}_x\mathbf{t}_{bi} + \mathbf{t}_x$$

où $\mathbf{R}_{ai} = \mathbf{R}_{i+1}\mathbf{R}_i^T$. Cette expression se réécrit sous forme matricielle en :

$$\begin{pmatrix}\mathbf{I}_3 - \mathbf{R}_{ai} & \mathbf{R}_{ai}\mathbf{t}_i & -\mathbf{t}_{i+1}\end{pmatrix}\begin{pmatrix}\mathbf{t}_x \\ \lambda_i \\ \lambda_{i+1}\end{pmatrix} = -\mathbf{R}_x\mathbf{t}_{bi}, \quad i = 1..n \tag{2.4}$$

Si l'on regroupe ces équations pour tous les i, on obtient un système dont la forme bloc est plaisante :

$$\begin{pmatrix}
\mathbf{I_R}_{a1} & \mathbf{R}_{a1}\mathbf{t}_1 & -\mathbf{t}_2 & 0 & 0 & \cdots & 0 \\
\mathbf{I_R}_{a2} & 0 & \mathbf{R}_{a2}\mathbf{t}_2 & -\mathbf{t}_3 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\mathbf{I_R}_{an-1} & 0 & \cdots & 0 & \mathbf{R}_{an-1}\mathbf{t}_{n-1} & -\mathbf{t}_n & 0 \\
\mathbf{I_R}_{an} & 0 & \cdots & 0 & 0 & \mathbf{R}_{an}\mathbf{t}_n & -\mathbf{t}_{n+1}
\end{pmatrix}\begin{pmatrix}\mathbf{t}_x \\ \lambda_1 \\ \vdots \\ \lambda_{n+1}\end{pmatrix} = \begin{pmatrix}-\mathbf{R}_x\mathbf{t}_{b1} \\ -\mathbf{R}_x\mathbf{t}_{b2} \\ \vdots \\ -\mathbf{R}_x\mathbf{t}_{bn-1} \\ -\mathbf{R}_x\mathbf{t}_{bn}\end{pmatrix} \tag{2.5}$$

Cette structure nous garantit que le système sera inversible dès lors que les $\mathbf{t}_i$ ne possèdent pas de coefficients nuls. De plus, si l'un des $\mathbf{t}_i$ possède un ou plusieurs coefficients nuls, il est fort probable que $\mathbf{R}_{ai}\mathbf{t}_i$ n'en possède pas. Par conséquent, ce système devrait être, en pratique, toujours inversible.

Ce système possède $3n$ équations et $(n+1)+3$ inconnues, il suffit donc de $n \geq 2$ mouvements de la caméra pour obtenir une solution. On n'augmente donc pas le nombre de mouvements minimal pour obtenir une solution par rapport aux méthodes classiques.

Enfin, on retrouve, ici aussi, le fait que des rotations pures du robot ne fournissent pas une solution unique.

2.4 Conclusion

Nous avons proposé dans ce chapitre une méthode d'auto-étalonnage pince-caméra, qui, à l'inverse des méthodes d'étalonnage pince-caméra existantes, fonctionne dans des scènes quelconques. Les résultats expérimentaux montrent une faible dégradation des résultats par rapport aux méthodes existantes au profit d'une plus grande autonomie. Cette autonomie est d'autant plus grande que la méthode est robuste aux petits mouvements qui garantissent la présence d'une scène commune dans toutes les images et donc la résolution du problème.

Nous avons aussi présenté une méthode dans le cas où la caméra observe un faisceau de droites concourantes. Elle permet de compenser la non observabilité de la profondeur du centre du faisceau par les mouvements du robot.

Chapitre 3

Étalonnage en ligne

3.1 Introduction

Au chapitre précédent, nous avons proposé une méthode pour effectuer un étalonnage pince-caméra sans utiliser de modèle de la scène observée par la caméra ou dans des cas particuliers où la profondeur de la scène ne peut être obtenue. Cette méthode permet d'augmenter l'autonomie des robots en leur permettant de s'auto-étalonner.

Une autre façon d'augmenter l'autonomie consiste à, non pas déterminer la transformation pince-caméra, mais à la garder à jour. En effet, en pratique, elle n'est pas forcément aussi constante qu'on le souhaite.

Par exemple, les vieux robots ont tendance à voir leur étalonnage interne varier. Ceci signifie que le centre du repère pince, qui est un point virtuel, ne se trouve plus physiquement au même endroit sur le robot. Or, le lien entre la caméra et la pince reste lui constant car ces deux objets physiques sont rigidement et mécaniquement liés. Par conséquent, le lien entre le repère virtuel de la pince et celui de la caméra varie aussi.

De façon duale, les conditions atmosphériques peuvent varier au cours de la mission du robot. Ceci influe sur le repère de la caméra et donc sur la transformation pince-caméra.

Pour pouvoir mettre à jour la transformation pince-caméra, nous utiliserons toujours la relation de conjugaison entre mouvement de la caméra et mouvement de la pince : $\mathbf{AX} = \mathbf{XB}$. Pour que cette relation reste valide, il faut que, pendant chaque mouvement, la transformation pince-caméra reste constante.

Comme nous avons supposé qu'elle ne l'était pas au cours du temps, il n'est pas possible d'assurer globalement cette condition. En revanche, on peut l'assurer localement en supposant que les variations de la transformation pince-caméra sont

lentes. Ceci implique de traiter chaque mouvement indépendamment de ceux qui le précèdent et de ceux qui lui succèdent. La condition de constance locale nous amène également à restreindre l'amplitude des déplacements du robot (*hypothèse des petits mouvements*).

Il n'est donc pas possible d'appliquer les méthodes des chapitres précédents. Il faut nous rabattre sur une solution itérative. Nous en proposons ici deux : la première est un filtre de Kalman, la seconde un filtre de Kalman étendu itératif (IEKF). Ces deux filtres sont déduits de la formulation linéaire vue au Chapitre 1 que nous rappelons ici.

Lorsqu'à l'instant k la caméra se déplace d'un déplacement rigide $\mathbf{A}_k = (\mathbf{R}_{Ak}, \mathbf{t}_{Ak})$ et le robot, sur lequel elle est fixée, d'un déplacement $\mathbf{B}_k = (\mathbf{R}_{Bk}, \mathbf{t}_{Bk})$, ces deux mouvements sont conjugués par la transformation pince-caméra $\mathbf{X}$. Ceci s'exprime de manière linéaire par :

$$\mathbf{C}_k \mathbf{x} = \mathbf{y}_k$$

où

$$\mathbf{C}_k = \begin{pmatrix} \mathbf{I}_9 - \mathbf{R}_{Ak} \otimes \mathbf{R}_{Bk} & \mathbf{0}_{9\times 3} \\ \mathbf{I}_3 \otimes (\mathbf{t}_{Bk}^T) & \mathbf{I}_3 - \mathbf{R}_{Ak} \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} vec(\mathbf{R}_X) \\ \mathbf{t}_X \end{pmatrix} \quad \mathbf{y}_k = \begin{pmatrix} \mathbf{0}_{9\times 1} \\ \mathbf{t}_{Ak} \end{pmatrix}$$

3.2 Filtre de Kalman

3.2.1 Modélisation

Filtre

Notons $\mathbf{x}_k$, la valeur de $\mathbf{x}$, à l'instant k. Ce vecteur est appelé *vecteur d'état* du système. L'estimation de ce vecteur à l'instant k en fonction des informations disponibles à l'instant $k-1$ sera notée $\mathbf{x}(k|k-1)$.

Déterminons à présent les deux équations du filtre de Kalman :

- l'équation d'état qui modélise l'évolution du vecteur d'état ;
- l'équation de mesure qui relie le vecteur d'état à ce qui peut être mesuré.

L'hypothèse de constance nous donne l'équation d'état du filtre :

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}(k) \tag{3.1}$$

où $\mathbf{v}(k)$ est un terme représentant les erreurs de modélisation, en particulier il permet de prendre en compte des variations lentes de la transformation pince-caméra.

Quant à l'équation de mesure du filtre, elle nous est fournie par la relation de conjugaison :

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{e}(k) \tag{3.2}$$

où $\mathbf{e}(k)$ représente les erreurs de mesure.

La solution de ce filtre est donnée par :

$$\begin{aligned}
\mathbf{P}_0 &= \mathbf{R}_0 && (3.3)\\
\mathbf{S}_k &= \mathbf{C}_k\mathbf{P}_k\mathbf{C}_k^T + \mathbf{R}_2 && (3.4)\\
\mathbf{K}_k &= \mathbf{P}_k\mathbf{C}_k^T\mathbf{S}_k^{-1} && (3.5)\\
\mathbf{P}_{k+1} &= \mathbf{P}_k + \mathbf{R}_1 - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T && (3.6)\\
\mathbf{x}(k+1|k) &= [\mathbf{I}_{12} - \mathbf{K}_k\mathbf{C}_k]\,\mathbf{x}(k|k-1) + \mathbf{K}_k\mathbf{y}_k && (3.7)
\end{aligned}$$

où $\mathbf{R}_0$ est une estimation initiale de $\mathbf{P}$ (que nous prendrons égale à $\mathbf{I}_{12}$), $\mathbf{R}_1$ est la matrice de covariance associée aux erreurs de modélisation $\mathbf{v}$, $\mathbf{R}_2$ est la matrice de covariance associée aux erreurs de mesure $\mathbf{e}$.

Comme nous supposons que chaque déplacement du système robot/caméra est de faible amplitude, la matrice $\mathbf{C}$ est mal conditionnée. Pour améliorer le comportement numérique, on peut alors ajouter une matrice de conditionnement Λ dans l'équation de mesure (3.2) qui devient :

$$\mathbf{y}'_k = \Lambda\big[\mathbf{C}_k\mathbf{x}_k + \mathbf{e}(k)\big]$$

Il faut alors insérer dans les équations, solution du filtre, les modifications appropriées.

Modélisation des erreurs

Dans les équations (3.1) et (3.2) du filtre de Kalman apparaissent des erreurs qu'il nous faut modéliser. Pour cela, nous allons faire les hypothèses suivantes :

- les erreurs $\mathbf{e}$ et $\mathbf{v}$ ne sont pas corrélées ;
- le mouvement du robot est parfait ;
- le mouvement de la caméra est mesuré à un bruit additif près. Cela ne respecte certes pas la contrainte d'orthonormalité des matrices de rotation, mais simplifie le modèle d'erreur.

La première hypothèse nous permet de modéliser les erreurs de modélisation (*sic !*) indépendamment de celles de mesure. Ainsi, on considère que $\mathbf{v}$ est un bruit Gaussien centré dont la matrice de covariance $\mathbf{R}_1$ est diagonale et de dimension (12×12).

Pour modéliser les erreurs de mesure, ajoutons un bruit additif au déplacement de la caméra :

$$\begin{aligned}
\mathbf{R}_{A_k}^{reel} &= \mathbf{R}_{A_k}^{mesure} + \Xi_A && (3.8)\\
\mathbf{t}_{A_k}^{reel} &= \mathbf{t}_{A_k}^{measure} + \mathbf{v}_A && (3.9)
\end{aligned}$$

Auto-étalonnage pince-caméra

où Ξ_A et $\mathbf{v}_A$ sont des bruits blancs (donc Gaussiens et centrés) non corrélés dont les matrices de covariance sont respectivement $\Gamma_1(9 \times 9)$ et $\Gamma_2(3 \times 3)$.

Reportons ce résultat dans les équations de conjugaison classiques (1.2) et (1.3) :

$$\begin{aligned} \mathbf{R}_X - \mathbf{R}_A^{mesure}\mathbf{R}_X\mathbf{R}_B^T &= \Xi_A\mathbf{R}_X\mathbf{R}_B^T \\ \mathbf{R}_X\mathbf{t}_B + \mathbf{t}_X - \mathbf{R}_A^{mesure}\mathbf{t}_X &= \mathbf{t}_A^{mesure} + \mathbf{v}_A + \Xi_A\mathbf{t}_X \end{aligned}$$

Pour les besoins de l'analyse, considérons dans un premier temps que $\mathbf{R}_X$ et $\mathbf{t}_X$ sont connus. Notons $\mathbf{v}'_A \stackrel{\Delta}{=} \mathbf{v}_A + \Xi_A\mathbf{t}_X$ et $\xi_{\mathbf{a}} = vec(\Xi_A\mathbf{R}_X\mathbf{R}_B^T)$. Avec ces notations, l'erreur de mesure s'écrit : $\mathbf{e} = ({\xi_{\mathbf{a}}}^T, \mathbf{v}_A'^T)^T$ et la matrice de covariance $\mathbf{R}_2$ qui lui est associée est donnée par le théorème suivant :

Théorème 8

Sous l'hypothèse que $\mathbf{R}_X$ et $\mathbf{t}_X$ sont connus et selon les hypothèses du modèle, la matrice de covariance $\mathbf{R}_2$ associée aux erreurs de mesure $\mathbf{e}$ est égale à :

$$\mathbf{R}_2 = \begin{pmatrix} \mathbf{U}^T\Gamma_1\mathbf{U} & \mathbf{U}^T\Gamma_1\mathbf{V} \\ \mathbf{V}^T\Gamma_1\mathbf{U} & \Gamma_2 + \mathbf{V}^T\Gamma_1\mathbf{V} \end{pmatrix}$$

$$\text{où } \mathbf{U} = \begin{pmatrix} \mathbf{R}_X{\mathbf{R}_B}_k^T & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{R}_X{\mathbf{R}_B}_k^T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{R}_X{\mathbf{R}_B}_k^T \end{pmatrix} \text{ et } \mathbf{V} = \begin{pmatrix} \mathbf{t}_X^T & \mathbf{0}_{3\times1}^T & \mathbf{0}_{3\times1}^T \\ \mathbf{0}_{3\times1}^T & \mathbf{t}_X^T & \mathbf{0}_{3\times1}^T \\ \mathbf{0}_{3\times1}^T & \mathbf{0}_{3\times1}^T & \mathbf{t}_X^T \end{pmatrix}$$

Preuve :

1. Calculons la covariance $V\left\{\xi_{\mathbf{a}i}, \xi_{\mathbf{a}j}\right\}$ pour tout couple $(i,j) \in \{1..9\}^2$.

Notons $(k,l) \in \{1,2,3\}^2$ et $(p,q) \in \{1,2,3\}^2$ les uniques couples, à (i,j) fixés, tels que $\xi_{\mathbf{a}i} = (\Xi_A\mathbf{R}_X\mathbf{R}_B^T)_{kl}$ et $\xi_{\mathbf{a}j} = (\Xi_A\mathbf{R}_X\mathbf{R}_B^T)_{pq}$.

Ces indices sont définis par $i = 3(k-1)+l$, $j = 3(p-1)+q$.

Alors,

$$\begin{aligned}
V\left\{\xi_{\mathbf{a}i},\xi_{\mathbf{a}j}\right\} &= V\left\{(\Xi_A\mathbf{R}_X\mathbf{R}_B^T)_{kl},(\Xi_A\mathbf{R}_X\mathbf{R}_B^T)_{pq}\right\}\\
&= V\left\{\sum_{r=1}^{3}(\Xi_A)_{kr}(\mathbf{R}_X\mathbf{R}_B^T)_{rl},\sum_{s=1}^{3}(\Xi_A)_{ps}(\mathbf{R}_X\mathbf{R}_B^T)_{sq}\right\}\\
&= \sum_{r=1}^{3}\left(\sum_{s=1}^{3}V\left\{(\Xi_A)_{kr},(\Xi_A)_{ps}\right\}(\mathbf{R}_X\mathbf{R}_B^T)_{sq}\right)(\mathbf{R}_X\mathbf{R}_B^T)_{rl}\\
&= \sum_{r=1}^{3}\left(\sum_{s=1}^{3}\Gamma_{1(3(k-1)+r,3(p-1)+s)}(\mathbf{R}_X\mathbf{R}_B^T)_{sq}\right)(\mathbf{R}_X\mathbf{R}_B^T)_{rl}\\
&= (\mathbf{U}^T\Gamma_1\mathbf{U})_{(3(k-1)+l,3(p-1)+q)}\\
&= (\mathbf{U}^T\Gamma_1\mathbf{U})_{ij}
\end{aligned}$$

2. Calculons la covariance $V\left\{\mathbf{v'}_{A_i},\mathbf{v'}_{A_j}\right\}$ pour tout couple $(i,j)\ \in\ \{1,2,3\}^2$

$$\begin{aligned}
V\left\{\mathbf{v}'_{A_i},\mathbf{v}'_{A_j}\right\} &= V\left\{(\mathbf{v}_A+\Xi_A\mathbf{t}_X)_i,(\mathbf{v}_A+\Xi_A\mathbf{t}_X)_j\right\}\\
&= V\left\{(\mathbf{v}_A)_i,(\mathbf{v}_A)_j\right\}+V\left\{(\mathbf{v}_A)_i,(\Xi_A\mathbf{t}_X)_j\right\}+\\
&\quad V\left\{(\Xi_A\mathbf{t}_X)_i,(\mathbf{v}_A)_j\right\}+V\left\{(\Xi_A\mathbf{t}_X)_i,(\Xi_A\mathbf{t}_X)_j\right\}\\
&= V\left\{(\mathbf{v}_A)_i,(\mathbf{v}_A)_j\right\}+V\left\{(\mathbf{v}_A)_i,(\Xi_A\mathbf{t}_X)_j\right\}+\\
&\quad V\left\{(\mathbf{v}_A)_j,(\Xi_A\mathbf{t}_X)_i\right\}+V\left\{(\Xi_A\mathbf{t}_X)_i,(\Xi_A\mathbf{t}_X)_j\right\}
\end{aligned}$$

Par définition, le premier terme $V\left\{(\mathbf{v}_A)_i,(\mathbf{v}_A)_j\right\}$ est égal à $(\Gamma_2)_{ij}$. Poursuivons nos calculs.

$$\begin{aligned}
V\left\{(\mathbf{v}_A)_i,(\Xi_A\mathbf{t}_X)_j\right\} &= V\left\{(\mathbf{v}_A)_i,\sum_{r=1}^{3}(\Xi_A)_{jr}(\mathbf{t}_X)_r\right\}\\
&= \sum_{r=1}^{3}\underbrace{V\left\{(\mathbf{v}_A)_i,(\Xi_A)_{jr}\right\}}_{0}(\mathbf{t}_X)_r\\
&= 0\\
V\left\{(\Xi_A\mathbf{t}_X)_i,(\Xi_A\mathbf{t}_X)_j\right\} &= V\left\{\sum_{r=1}^{3}(\Xi_A)_{ir}(\mathbf{t}_X)_r,\sum_{s=1}^{3}(\Xi_A)_{js}(\mathbf{t}_X)_s\right\}\\
&= \sum_{r=1}^{3}\left(\sum_{s=1}^{3}V\left\{(\Xi_A)_{ir},(\Xi_A)_{js}\right\}(\mathbf{t}_X)_s\right)(\mathbf{t}_X)_r\\
&= (\mathbf{V}^T\Gamma_1\mathbf{V})_{ij}
\end{aligned}$$

On obtient donc finalement :

$$V\left\{\mathbf{v}'_{A_i}, \mathbf{v}'_{A_j}\right\} = (\Gamma_2 + \mathbf{V}^T\Gamma_1\mathbf{V})_{ij}$$

3. Calculons la covariance $V\left\{\mathbf{v}'_{A_i}, \xi_{\mathbf{a}j}\right\}$ pour tout couple (i,j), $i = 1..3$ et $j = 1..9$.

Notons $(k,l) \in \{1,2,3\}^2$, l'unique couple , à j fixé, tel que $\xi_{\mathbf{a}j} = (\Xi_A\mathbf{R}_X\mathbf{R}_B^T)_{kl}$ $(j = 3(k-1)+l)$.

$$\begin{aligned}
V\left\{\mathbf{v}'_{A_i}, \xi_{\mathbf{a}j}\right\} &= V\left\{(\mathbf{v}_A + \Xi_A\mathbf{t}_X)_i, (\Xi_A\mathbf{R}_X\mathbf{R}_B^T)_{kl}\right\} \\
&= V\left\{(\mathbf{v}_A)_i, (\Xi_A\mathbf{R}_X\mathbf{R}_B^T)_{kl}\right\} + V\left\{(\Xi_A\mathbf{t}_X)_i, (\Xi_A\mathbf{R}_X\mathbf{R}_B^T)_{kl}\right\} \\
&= V\left\{(\mathbf{v}_A)_i, \sum_{r=1}^{3}(\Xi_A)_{kr}(\mathbf{R}_X\mathbf{R}_B^T)_{rl}\right\} + \\
&\quad V\left\{\sum_{s=1}^{3}(\Xi_A)_{is}(\mathbf{t}_X)_s, \sum_{r=1}^{3}(\Xi_A)_{kr}(\mathbf{R}_X\mathbf{R}_B^T)_{rl}\right\} \\
&= \sum_{r=1}^{3}\underbrace{V\left\{(\mathbf{v}_A)_i, (\Xi_A)_{kr}\right\}}_{0}(\mathbf{R}_X\mathbf{R}_B^T)_{rl} + \\
&\quad \sum_{s=1}^{3}\left(\sum_{r=1}^{3}V\left\{(\Xi_A)_{is}, (\Xi_A)_{kr}\right\}(\mathbf{R}_X\mathbf{R}_B^T)_{rl}\right)(\mathbf{t}_X)_s \\
&= (\mathbf{V}^T\Gamma_1\mathbf{U})_{ij}
\end{aligned}$$

□

Comme cette expression de $\mathbf{R}_2$ dépend de quantités inconnues ($\mathbf{R}_X$ et $\mathbf{t}_X$) ou variables ($\mathbf{R}_{Bk}$), nous sommes amenés à faire des approximations supplémentaires :

$$\begin{aligned}
\mathbf{R}_X &\approx \mathbf{I}_3 \\
\mathbf{t}_X &\approx \mathbf{0}_{3\times 1} \\
\mathbf{R}_B &\approx \mathbf{I}_3
\end{aligned}$$

Alors, $\mathbf{R}_2$ devient : $\mathbf{R}_2 = \begin{pmatrix} \Gamma_1 & \mathbf{0}_{9\times 3} \\ \mathbf{0}_{3\times 9} & \Gamma_2 \end{pmatrix}$.

Comme les matrices Γ_1 et Γ_2 ont été choisies diagonales, $\mathbf{R}_2$ est donc elle aussi diagonale. On peut alors considérer que les coefficients du vecteur de mesure sont des scalaires indépendants, ce qui permet de réduire le temps de calcul et d'augmenter le conditionnement numérique du filtre [GA93, pp. 115–116].

3.2.2 Du vecteur d'état à une transformation rigide

Après chaque mouvement, le filtre précédent détermine un nouveau vecteur d'état $\mathbf{x}_k$. Ce vecteur d'état contient l'estimation courante de la matrice de rotation pince-caméra $\mathbf{R}_X$, sous la forme de l'estimation de ses trois vecteurs colonnes ($\mathbf{i}$, $\mathbf{j}$ et $\mathbf{k}$).

Cependant, le filtre ne garantit pas que l'estimation de ces trois vecteurs forme une base orthonormée directe. Il n'est donc pas certain que l'estimation de $\mathbf{R}_X$ par le filtre soit bien une matrice de rotation.

Il est donc nécessaire de transformer ces trois vecteurs en une matrice de rotation. Deux questions se posent alors : « Quand faire cette transformation ? » et « Comment la faire au mieux numériquement ? ».

Comment le faire efficacement ?

L'opération consistant à trouver la matrice de rotation $\mathbf{R}$ qui s'accorde le mieux aux vecteurs colonnes $\mathbf{i}, \mathbf{j}, \mathbf{k}$ s'appelle *orthonormalisation*.

La méthode classique d'orthonormalisation est la méthode de Gram-Schmidt : à partir de trois vecteurs $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, de $\mathbb{R}^3$, on obtient une base orthonormée $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ par

$$\begin{aligned} \mathbf{w}_1 &= \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} \\ \mathbf{w}_2 &= \frac{\mathbf{v}_2 - (\mathbf{v}_2^T \mathbf{w}_1)\mathbf{w}_1}{\|\mathbf{v}_2 - (\mathbf{v}_2^T \mathbf{w}_1)\mathbf{w}_1\|} \\ \mathbf{w}_3 &= \frac{\mathbf{v}_3 - (\mathbf{v}_3^T \mathbf{w}_1)\mathbf{w}_1 - (\mathbf{v}_3^T \mathbf{w}_2)\mathbf{w}_2}{\|\mathbf{v}_3 - (\mathbf{v}_3^T \mathbf{w}_1)\mathbf{w}_1 - (\mathbf{v}_3^T \mathbf{w}_2)\mathbf{w}_2\|} = \mathbf{w}_1 \times \mathbf{w}_2 \end{aligned} \tag{3.10}$$

Cependant, cette méthode n'a aucun intérêt pratique. En effet, elle dépend de l'ordre dans lequel on traite les vecteurs et elle construit les vecteurs de la base orthonormée les uns après les autres, les suivants dépendant des précédents. Le résultat numérique est désastreux.

Il existe, à notre connaissance, trois méthodes numériquement stables : une méthode basée sur une décomposition en valeurs singulières (SVD) [PTVF92], une autre utilisant la décomposition QR [PTVF92] et une méthode de minimisation dans l'espace des quaternions [HCDL95, Chr98].

Une matrice $\mathbf{M}$ se décompose en :

$$\begin{aligned} \mathbf{M} &= \mathbf{U}\Sigma\mathbf{V}^T \quad (SVD) \\ &= \mathbf{N}\mathbf{P} \quad (QR) \end{aligned}$$

où les matrices $\mathbf{U}$,$\mathbf{V}$ et $\mathbf{N}$ sont des matrices orthonormales, Σ est une matrice diagonale à coefficients positifs ou nuls et $\mathbf{P}$ est une matrice triangulaire supérieure [1].

1. J'ai volontairement évité d'écrire la décomposition QR de $\mathbf{M}$ sous la forme classique $\mathbf{M} = \mathbf{QR}$ puisque, sous cette forme, c'est $\mathbf{Q}$ et non $\mathbf{R}$ qui est une matrice de rotation, ce qui peut prêter à confusion

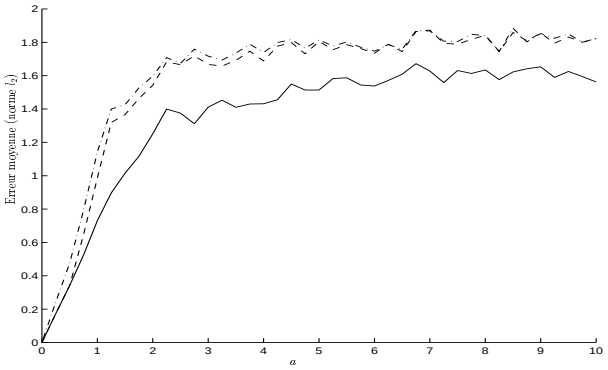


FIG. 3.1: *Sensibilité au bruit des méthodes d'orthonormalisation suivante : SVD (tirets), décomposition QR (trait mixte) et quaternions (trait plein).*

On peut alors approcher $\mathbf{M} = (\mathbf{i}, \mathbf{j}, \mathbf{k})$ par les matrices orthonormales suivantes :

$$\begin{aligned} \mathbf{M} &\approx \mathbf{U}\mathbf{V}^T \\ &\approx \mathbf{P} \end{aligned}$$

La troisième méthode cherche la matrice de rotation $\mathbf{R}$ qui satisfait « au mieux » :

$$\mathbf{R}\begin{pmatrix}\mathbf{i} & \mathbf{j} & \mathbf{k}\end{pmatrix}^T = \mathbf{I}_{3\times 3}$$

Ce problème se ramène à un problème de minimisation dans l'espace des quaternions de la forme :

$$\min_{\mathbf{q}}(\mathbf{q}^T\mathbf{B}\mathbf{q})$$

où $\mathbf{B}$ est une matrice qui dépend des vecteurs $\mathbf{i}, \mathbf{j}, \mathbf{k}$. Nous ne détaillerons pas plus avant cette méthode.

Nous avons comparé numériquement ces trois méthodes de la façon suivante. Cinq cents matrices de rotation de référence ont été tirées aléatoirement. Ces matrices ont été bruitées par l'ajout d'un bruit uniformément réparti sur $[-a, a]$, a variant. Nous avons appliqué les trois méthodes précédentes à chacune de ces matrices bruitées et calculé la norme matricielle l_2 de la différence entre la matrice de référence et la matrice reconstruite. La Figure 3.1 montre la moyenne de ces erreurs en fonction du niveau de bruit considéré. Elle incite donc à considérer la méthode des quaternions comme la plus efficace numériquement.

Quand?

Il y a trois façons de procéder (voir Figure 3.2):

a) On peut se contenter de réordonner les vecteurs $\mathbf{i}, \mathbf{j}$ et $\mathbf{k}$, tels qu'ils ont été estimés, dans une matrice $\mathbf{R}_X$ qui ne satisfasse pas la condition d'orthogonalité ;

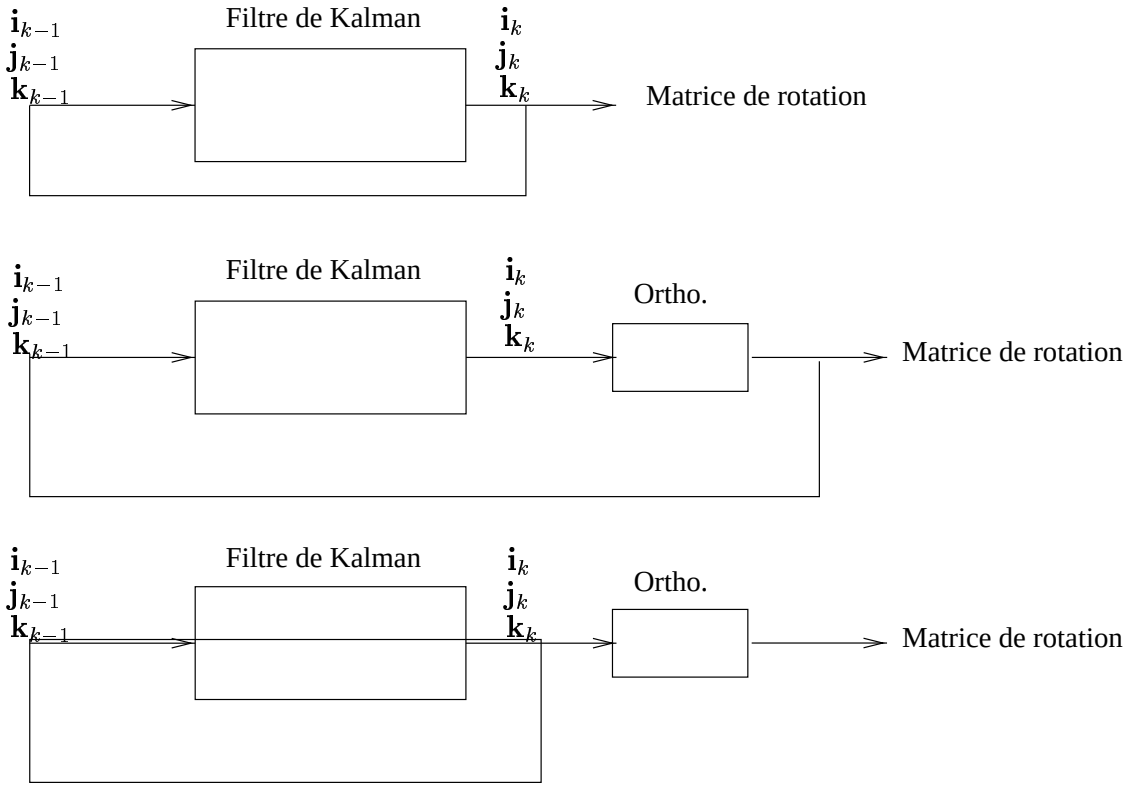


FIG. 3.2: *Représentation schématique des trois façons d'utiliser le vecteur d'état.. De haut en bas : sans orthonormalisation, orthonormalisation interne et orthonormalisation externe*

b) On peut appliquer une méthode d'orthonormalisation à l'intérieur du filtre, en assurant à chaque instant que le vecteur d'état corresponde à une transformation rigide (*orthonormalisation interne*) ;

c) On peut appliquer la méthode d'orthonormalisation à une copie du vecteur d'état, laissant celui-ci inchangé (*orthonormalisation externe*).

Pour comparer ces trois méthodes, nous avons fait cinquante simulations du filtre avec à chaque simulation, un tirage aléatoire différent d'une transformation pince-caméra et des mouvements de caméra aléatoires. Pour chaque simulation, nous avons lancé le filtre avec chacun des trois types de traitement et mesuré l'évolution de l'erreur entre la matrice de rotation estimée et la matrice de rotation de référence. La Figure 3.3 montre l'évolution de l'erreur moyenne.

La première chose que l'on remarque est que le fait d'appliquer une orthonormalisation réduit l'erreur dès les premières itérations, contrairement au cas où aucun traitement n'est effectué. La seconde est que la méthode d'orthonormalisation externe est celle qui permet de converger le plus vite. Enfin, chose plus surprenante à

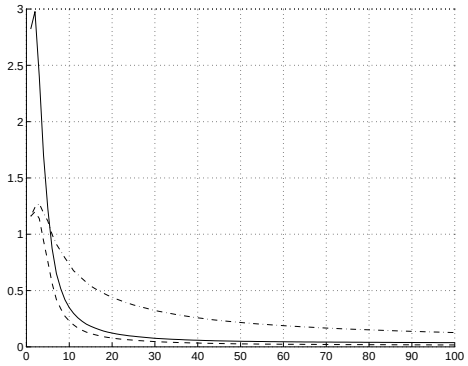


FIG. 3.3: *Évolution de l'erreur moyenne entre la matrice de rotation estimée et la matrice de rotation réelle* **a)** *sans orthonormalisation (trait plein),* **b)** *orthonormalisation interne (trait mixte),* **c)** *orthonormalisation externe (pointillés).*

première vue, la méthode d'orthonormalisation interne est la plus lente à converger. Ceci s'explique en fait par l'interprétation du procédé d'orthonormalisation comme une perturbation du filtre qui tire le filtre hors de sa convergence optimale.

Nous concluons donc qu'il faut faire la transformation du vecteur d'état en un déplacement rigide à l'extérieur du filtre.

3.3 Filtre de Kalman étendu itératif[2]

Le filtre précédent suppose que tous les coefficients du vecteur d'état $\mathbf{x}_k$ sont indépendants et ne tient donc pas compte des contraintes non linéaires d'orthogonalité. Pour pouvoir utiliser les résultats de ce filtre, il faut donc passer les 9 premiers coefficients de ce vecteur d'état à travers un opérateur d'orthonormalisation.

Plutôt que d'utiliser ces deux étapes de minimisation successives, on peut utiliser un filtre de Kalman étendu(EKF), conçu pour tirer parti des non linéarités. Cependant, un tel filtre est connu pour être peu stable et il est préférable d'en utiliser une version itérative : le filtre de Kalman étendu itératif (IEKF).

Nous allons déterminer ce filtre en commençant par déterminer un nouveau vecteur d'état contenant les paramètres les plus pertinents possibles du système (1.6).

Dans le but de calculer la pose d'un objet 3D, Lowe [Low87] estime itérativement une matrice de rotation $\mathbf{R}$. Pour cela, il conserve la structure matricielle qu'il met à jour, à chaque itération, en prémultipliant l'estimation courante $\tilde{\mathbf{R}}$ de $\mathbf{R}$ par une matrice de correction $\delta\mathbf{R}$. Celle-ci est une matrice de rotation autour des axes x,

2. Ce travail a été effectué lors d'un séjour à l'*Institut für Algorithmen und Kognitive Systeme (*IAKS*)* du *Fraunhofer-Institut für Informations- und Datenverarbeitung (*IITB*)* de Karlsruhe, sous la direction du Professeur H.-H. Nagel.

y et z courants et dont les angles ϕ_x, ϕ_y et ϕ_z sont petits. On peut alors utiliser l'hypothèse des petits mouvements pour simplifier les calculs, puisqu'alors les angles ϕ_x, ϕ_y et ϕ_z peuvent être considérés comme indépendants.

Similairement, nous pouvons définir un filtre de Kalman étendu itératif qui estime les corrections, supposées petites, à apporter à une estimation $\tilde{\mathbf{R}}$ de la rotation pince-caméra $\mathbf{R}_x$ plutôt que cette dernière. Notre nouveau vecteur d'état est donc

$$\mathbf{z} = (\phi_x, \phi_y, \phi_z, {\mathbf{t}_x}^T)^T. \tag{3.11}$$

Comme la transformation pince-caméra est censée être constante, nous supposons que les trois angles de correction sont nuls quelle que soit leur valeur précédente. De plus, $\mathbf{t}_x$ est alors supposée constante aux erreurs numériques près. On aboutit alors à l'équation d'état suivante :

$$\mathbf{z}_{k+1} = \mathbf{M}\mathbf{z}_k + \mathbf{w}(k), \quad \mathbf{M} = \begin{pmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 \end{pmatrix} \tag{3.12}$$

où $\mathbf{w}(k)$ est un bruit Gaussien de moyenne nulle et de matrice de covariance $\Gamma_{\mathbf{W}}$.

Cette matrice de covariance $\Gamma_{\mathbf{W}}$ est prise diagonale. En effet, nous nous plaçons sous l'hypothèse que les angles ϕ_x, ϕ_y et ϕ_z sont petits. Selon Lowe, ils peuvent alors être considérés comme indépendants. Nous faisons alors l'hypothèse supplémentaire que les erreurs que l'on commet dans leur estimation sont aussi indépendantes. De plus, nous supposerons que les bruits associés à chaque composante de $\mathbf{t}_x$ sont mutuellement décorrélés. Enfin, en considérant que le bruit associé aux angles de correction et celui associé à $\mathbf{t}_x$ ne sont pas corrélés, nous pouvons prendre $\Gamma_{\mathbf{W}}$ diagonale.

Cherchons à présent l'équation de mesure de ce filtre. Pour cela, nous allons exprimer $\mathbf{x}_k$ à partir de $\mathbf{z}_k$. Faisons d'abord apparaître dans $\mathbf{x}_k$ la matrice de correction $\delta\mathbf{R}_k$ et l'estimation courante de $\mathbf{R}_x$.

$$\mathbf{x}_k = \begin{pmatrix} vec(\mathbf{R}_{xk}) \\ \mathbf{t}_{xk} \end{pmatrix} = \begin{pmatrix} vec(\delta\mathbf{R}_k\tilde{\mathbf{R}}_k) \\ \mathbf{t}_{xk} \end{pmatrix}$$

En utilisant les propriétés de l'opérateur *vec*, nous obtenons

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{I}_3 \otimes \tilde{\mathbf{R}}_k \; vec(\delta\mathbf{R}_k) \\ \mathbf{t}_{xk} \end{pmatrix}$$

Si l'on note

$$f(\mathbf{z}_k) = \begin{pmatrix} vec(\delta\mathbf{R}_k) \\ \mathbf{t}_{xk} \end{pmatrix} \quad \text{et} \quad \mathbf{D}_k = \begin{pmatrix} \mathbf{I}_3 \otimes \tilde{\mathbf{R}}_k & \mathbf{0}_{9\times3} \\ \mathbf{0}_{3\times9} & \mathbf{I}_3 \end{pmatrix}$$

nous pouvons reporter $\mathbf{x}_k$ dans (3.2) et obtenir l'équation de mesure suivante :

$$\mathbf{y}_k = \mathbf{C}_k\mathbf{D}_k f(\mathbf{z}_k) + \mathbf{e}(k) \tag{3.13}$$

Auto-étalonnage pince-caméra

où l'on remarque que le vecteur $\mathbf{e}(k)$ est identique à celui qui figure dans (3.2). Il sera donc Gaussien, de moyenne nulle et de matrice de covariance $\mathbf{R}_2$ définie au Théorème 8.

Le filtre de Kalman étendu, construit à partir de (3.12) et (3.13) selon les formules données dans [BSF88], est alors défini par les équations intermédiaires suivantes :

$$\hat{\mathbf{z}}_{(k+1|k)} = \mathbf{M}\ \hat{\mathbf{z}}_{(k|k)} \tag{3.14}$$
$$\mathbf{P}_{(k+1|k)} = \mathbf{M}\mathbf{P}_{(k|k)}\mathbf{M}^T + \mathbf{R}_2 \tag{3.15}$$
$$\mathbf{J}_{k+1} = \mathbf{C}_k\mathbf{D}_k\frac{\partial f}{\partial \mathbf{z}}(\hat{\mathbf{z}}_{(k|k)}) \tag{3.16}$$
$$\mathbf{S}_{k+1} = \mathbf{J}_{k+1}\mathbf{P}_{(k+1|k)}\mathbf{J}_{k+1}^T + \Gamma_{\mathbf{W}} \tag{3.17}$$
$$\mathbf{K}_{k+1} = \mathbf{P}_{(k+1|k)}\mathbf{J}_{k+1}^T\mathbf{S}_{k+1}^{-1} \tag{3.18}$$

et les équations de mise à jour :

$$\hat{\mathbf{z}}_{(k+1|k+1)} = \mathbf{K}_{k+1}\mathbf{y}_{k+1} + (\mathbf{I}_6 - \mathbf{K}_{k+1}\mathbf{J}_{k+1})\hat{\mathbf{z}}_{(k+1|k)} \tag{3.19}$$
$$\mathbf{P}_{(k+1|k+1)} = \mathbf{P}_{(k+1|k)} - \mathbf{K}_{k+1}\mathbf{S}_{k+1}\mathbf{K}_{k+1}^T \tag{3.20}$$
$$\tilde{\mathbf{R}}_{k+1} = \delta\mathbf{R}(\hat{\mathbf{z}}_{(k+1|k+1)})\tilde{\mathbf{R}}_k \tag{3.21}$$

Dans les équations précédentes, seul le Jacobien de f ($\frac{\partial f}{\partial \mathbf{z}}$) n'est pas encore défini. Il est relié aux dérivées partielles de $\delta\mathbf{R}$ par rapport à chacun de ses angles par :

$$\frac{\partial f}{\partial \mathbf{z}} = \begin{pmatrix} \left(vec(\frac{\partial \delta\mathbf{R}}{\partial \theta})\right)_{\theta=\phi_x,\phi_y,\phi_z} & \mathbf{0}_{9\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{pmatrix}$$

Pour calculer ce nouveau Jacobien ($\frac{\partial \delta\mathbf{R}}{\partial \theta}$), on utilise l'hypothèse des petits angles qui permet d'écrire :

$$\delta\mathbf{R} = \begin{pmatrix} 1 & -\phi_z & \phi_y \\ \phi_z & 1 & -\phi_x \\ -\phi_y & \phi_x & 1 \end{pmatrix}$$

et d'obtenir

$$\frac{\partial \delta\mathbf{R}}{\partial \phi_x} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad \frac{\partial \delta\mathbf{R}}{\partial \phi_y} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad \frac{\partial \delta\mathbf{R}}{\partial \phi_z} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

On obtient donc au final :

$$\frac{\partial f}{\partial \mathbf{z}} = \begin{pmatrix} As(1,0,0) & \mathbf{0}_{3\times 3} \\ As(0,1,0) & \mathbf{0}_{3\times 3} \\ As(0,0,1) & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{pmatrix}$$

Le filtre de Kalman étendu itératif est obtenu, toujours en adaptant les formules données par [BSF88], en remplaçant les équations de mise à jour (3.19)–(3.21) par une procédure itérative :

- $\hat{\mathbf{z}}^0_{(k+1|k+1)} = \hat{\mathbf{z}}_{(k+1|k)}$
- $\tilde{\mathbf{R}}^0_k = \tilde{\mathbf{R}}_k$
- $\mathbf{D}^i_k = \mathbf{D}_k$
- Répéter jusqu'à convergence
  - $\mathbf{H}_i = \mathbf{C}_k \mathbf{D}^i_k \frac{\partial f}{\partial \mathbf{z}}(\hat{\mathbf{z}}^i_{(k+1|k+1)})$
  - $\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_{(k+1|k)} \mathbf{H}_i^T + \Gamma_{\mathbf{W}}$
  - $\mathbf{K}_i = \mathbf{P}_{(k+1|k)} \mathbf{H}_i^T \mathbf{S}_i^{-1}$
  - $\mathbf{P}^i = (\mathbf{I}_6 - \mathbf{K}_i \mathbf{H}_i) \mathbf{P}_{(k+1|k)}$
  - $\hat{\mathbf{z}}^{i+1}_{(k+1|k+1)} = \hat{\mathbf{z}}_{(k+1|k)} + \mathbf{P}^i \mathbf{H}_i^T \Gamma_{\mathbf{W}}^{-1} (\mathbf{y}_{k+1} - \mathbf{C}_k \mathbf{D}^i_k f(\hat{\mathbf{z}}_{(k+1|k)}))$
  - $\tilde{\mathbf{R}}^{i+1}_k = \delta\mathbf{R}(\hat{\mathbf{z}}^{i+1}_{(k+1|k+1)}) \tilde{\mathbf{R}}^i_k$
  - Mettre à jour $\mathbf{D}^{i+1}_k$ avec $\tilde{\mathbf{R}}^{i+1}_k$
- $\hat{\mathbf{z}}_{(k+1|k+1)} = \hat{\mathbf{z}}^i_{(k+1|k+1)}$
- $\mathbf{P}_{(k+1|k+1)} = \mathbf{P}^i$
- $\tilde{\mathbf{R}}_{k+1} = \tilde{\mathbf{R}}^i_k$

3.4 Résultats expérimentaux

Dans le choix du vecteur d'état du filtre IEKF, nous avons suivi la préférence de Lowe pour l'utilisation d'angles de la matrice de correction à celle des angles de la matrice de rotation Toutefois, un filtre IEKF peut être défini à partir de ces derniers. Nous ne l'avons pas présenté ici car, ainsi que nous allons le voir, il ne fournit pas de bons résultats.

Nous avons comparé le filtre de Kalman et sa version étendue itérative, proposés dans les paragraphes précédents, ainsi que le filtre IEKF basé sur les angles de rotations. Pour cela, nous avons enregistré une séquence de mouvements. Avec cette séquence, nous avons calculé la transformation pince-caméra avec la méthode linéaire du chapitre 1. Le résultat obtenu a été pris comme référence. Puis, nous avons appliqué chacun des trois filtres à cette même séquence et calculé après chaque itération du filtre l'erreur entre son estimation courante de la transformation pince-caméra et la transformation de référence. La Figure 3.4 montre l'évolution de ces erreurs.

La courbe des erreurs en rotation montre que la préférence de Lowe reste bien fondée dans notre cas, puisque l'erreur en rotation du filtre IEKF à base d'angles de rotation « explose ».

La courbe des erreurs en translation montre l'amélioration obtenue en utilisant le filtre IEKF à base d'angles de correction plutôt que le filtre de Kalman.

Toutefois, ces résultats ont été obtenus après de longs tâtonnements pour trouver les bonnes matrices de covariance et avec une assez bonne estimation initiale. Ceci laisse donc à penser qu'il doit être possible, voire nécessaire, d'améliorer le comportement numérique de ces méthodes.

3.5 Conclusion

Ce chapitre présentait une première tentative d'étalonnage pince-caméra en ligne : après chaque mouvement du système pince/caméra, une estimation de la transformation pince-caméra est mise à jour à partir de ce dernier mouvement. Pour cela, nous avons défini un filtre de Kalman et un filtre de Kalman étendu itératif.

Les résultats expérimentaux sont encourageants et suggèrent une étude plus poussée de l'étalonnage pince-caméra en ligne. Toutefois, l'utilisation des filtres définis ici s'avère très sensible au choix des matrices de covariance. Cette sensibilité doit pouvoir être réduite par une modélisation plus fine du bruit intervenant dans les équations des différents filtres.

Enfin, on note que les équations des deux filtres incluent l'inversion d'une matrice $\mathbf{S}_k$ qui, du fait de la singularité de $\mathbf{C}_k$, n'est inversible que grâce à la présence de bruit. Ceci s'explique par le fait que la relation de conjugaison entre déplacement de la pince et déplacement de la caméra offre une infinité de solutions.

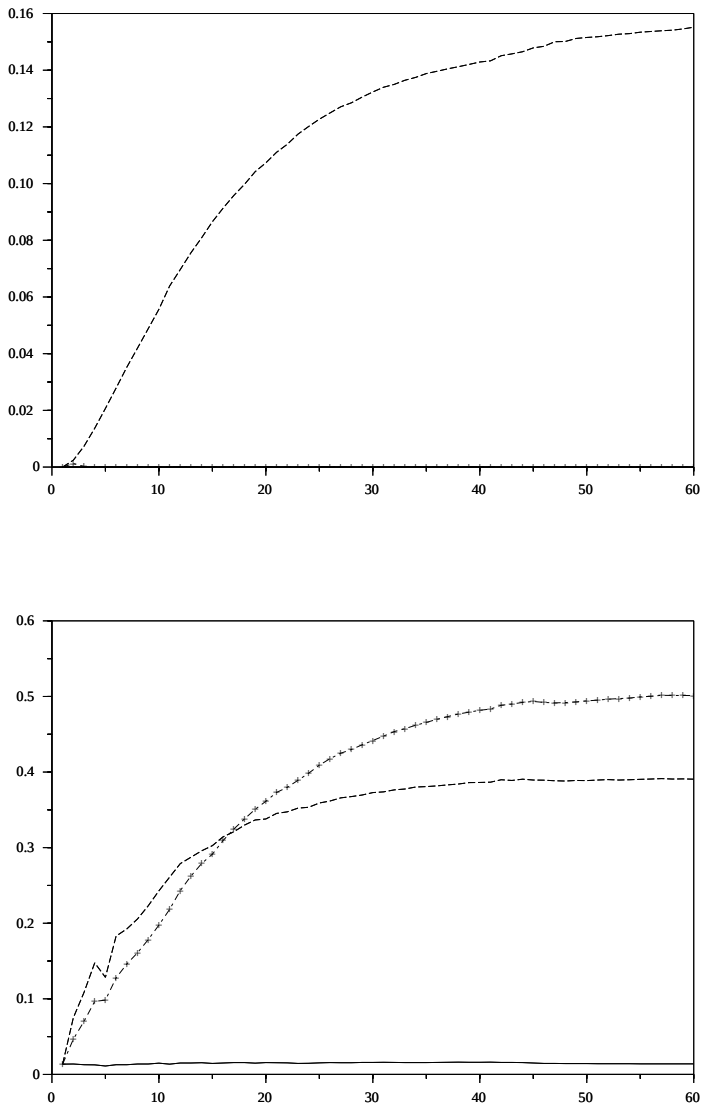


FIG. 3.4: *Évolution de l'erreur en rotation (en haut) et en translation (en bas) entre la transformation pince-caméra de référence et son estimation courante par chacun des filtres considérés : filtre de Kalman (trait mixte), IEKF à base d'angles de correction (trait plein) et IEKF à base d'angles de rotation (tirets).*

Dans les chapitres précédents, nous disposions de plusieurs mouvements, qui permettaient d'obtenir une solution unique, intersection des espaces des solutions proposés par chaque mouvement.

Il serait donc intéressant, à l'avenir, de conserver l'historique de ces espaces de solutions et d'en tenir compte dans les filtres – ou tout autre méthode itérative – pour réduire la dimension de l'espace des solutions proposé après chaque nouveau mouvement.

Conclusion et perspectives

Fromage d'Allobrogie

Dauphiné	**Savoie**
Saint-Marcellin	Abondance
Bleu de Sassenage	Vacherin d'Abondance
Rivieron	Reblochon frais de Féternes
	Sérac de chèvre

À déguster avec du Ripaille.

Résultats obtenus

À l'issu de cette thèse, nous avons dépoussiéré l'utilisation de droites dans les images pour l'asservissement visuel. Pour cela, nous avons montré que les coordonnées de Plücker des droites étaient sous-employées dans cette discipline. Nous en avons déduit un alignement en coordonnées de Plücker binormées, qui regroupe des informations purement 2D et des informations 3D. Cela nous a permis de définir *analytiquement* deux variantes d'une commande en deux phases partiellement découplée en rotation et translation (commande CPN). Ces deux commandes ont ainsi pu faire l'objet d'une étude de convergence qui s'est avérée positive. Les résultats expérimentaux ont néanmoins confirmé la nécessité, entrevue dans la partie théorique, de fusionner les deux phases de la commande.

Dans une deuxième partie, nous avons reformulé le problème de l'étalonnage pince-caméra sous forme linéaire. Cela nous a permis de faire une étude algébrique claire du problème. Nous avons ainsi obtenu une vue synthétique complète des résultats dispersés dans la littérature. La formulation linéaire du problème nous a ensuite permis de définir une méthode d'auto-étalonnage pince-caméra libérant des contraintes pratiques imposées par les méthodes classiques d'étalonnage. Enfin, toujours sur la base de la formulation linéaire du problème, nous avons proposé une méthode d'étalonnage en ligne qui permet de tenir compte de variations lentes de la transformation pince-caméra.

Perspectives

À court terme, nos travaux peuvent être poursuivis dans cinq directions. La première consiste à étendre les résultats de convergence de la commande CPN. Cette voie semble difficile mais sera pourtant facilitée par les résultats récents en matière de systèmes cascadés [PL98]. Une deuxième voie, aussi ardue à première vue, concerne l'étude de la robustesse de cette commande aux erreurs de mesure et d'étalonnage. Une troisième voie, plus facile, est d'étendre cette loi de commande au cas stéréoscopique étalonné. On pourrait ainsi se passer de modèle de la scène pour calculer les orientations nécessaires à la commande. Du côté de l'auto-étalonnage pince-caméra se profilent deux destinations : l'auto-étalonnage en ligne est la fusion de l'auto-étalonnage et de l'étalonnage en ligne présentés ici ; la prise en compte de transformations pince-caméra variables, par exemple dans le cas d'yeux motorisés, est aussi à envisager.

À moyen terme, on peut envisager d'effectuer le calcul des orientations par reconstruction euclidienne, extension du cas stéréo envisagé ci-dessus. Dans le même ordre d'idée, on relâchera les contraintes d'étalonnage pour aboutir à un asservissement non étalonné. D'un point de vue théorique, et même si cela laisse présager des difficultés numériques, on étendra l'utilisation des droites par celles des courbes

en s'intéressant aux tangentes de ces dernières. Enfin, il faudra un jour répondre à la question qui hante les chercheurs en matière d'étalonnage pince-caméra, à savoir quels en sont les mouvements optimaux.

Enfin, à plus long terme, il faudra pouvoir travailler sur des scènes complexes. Par exemple, dans le cadre des robots industriels, où l'on possède des modèles de l'environnement du robot, il pourra être intéressant de rapprocher l'asservissement visuel des techniques de fusion de données. En effet, les modèles sont souvent imprécis et les pures informations visuelles bruitées, il n'est donc pas toujours possible de travailler avec un seul type d'information. En dépassant le cadre restreint de cette thèse, il semble aussi important de poursuivre les travaux concernant l'asservissement visuel de robots non holonomes, entamés par Tsakiris *et al.* [TRS97]. En effet, de nombreuses applications s'ouvrent à de tels travaux, tels que les véhicules automobiles automatisés ou les robots à pattes, notamment le bipède de l'équipe Bip. Une des nombreuses tâches émergentes en robotique étant l'exploration, il sera nécessaire de pouvoir choisir ou détecter un objet (connu ou non) dans une scène de manière automatique puis de générer, toujours automatiquement, une consigne dans l'image permettant, par exemple, de s'approcher de l'objet.

Annexes

Rézules de poires

Pour 6 personnes
Ingrédients : 250g de farine, 150g de beurre, 2kg de poires à cuire, 400g de sucre, cannelle, sel

- Épluchez les poires. Coupez-les en quartiers et faites-les cuire à feu doux dans une grande casserole avec un fond d'eau, le sucre et un peu de cannelle, pendant 1h30 environ avec un couvercle. Laissez refroidir.

- Préparez la pâte brisée : sur une planche de bois, versez la farine avec une pincée de sel. Coupez le beurre en petits morceaux et incorporez-le à la farine en l'écrasant avec le bout des doigts. Faites un puits, versez un peu d'eau pour former une boule de pâte. Pétrissez le moins possible pour éviter qu'elle devienne élastique. Laissez reposer jusqu'au moment de la cuisson.

- Abaissez la pâte au rouleau et, avec un bol, découpez des ronds. Déposez une cuillère à soupe de poires cuites, humectez le bord du rond avec un peu d'eau et repliez en soudant bien avec les doigts. Gardez-les ainsi jusqu'au moment de les servir, car elles doivent être cuites au dernier moment.

- Plongez les rézules dans un grand bain d'huile chaude jusqu'à obtention d'une belle couleur dorée. Servez tout de suite.

- Vous pouvez saupoudrer les rézules de sucre glace.

Recette tirée de *Les Alpes, Carnet de cuisine*, Romain Pages Éditions, 1996.

Annexe A

Géométrie des droites

Rappelons en premier lieu, qu'une droite de l'espace physique possède quatre degrés de liberté. Autrement dit, elle est définie par quatre paramètres indépendants. Pour s'en rendre compte, on peut considérer deux plans parallèles (par exemple, $z = 0$ et $z = 1$) et une droite non parallèle à ces plans (Figure A.1). Alors, la droite est définie de manière unique par ses deux points d'intersection, $\boldsymbol{P}_1$ et $\boldsymbol{P}_2$, avec les plans. Chacun de ces deux points est repéré par deux coordonnées dans le plan auquel il appartient (dans notre exemple, $\boldsymbol{P}_1 = (x_1, y_1, 0)$ et $\boldsymbol{P}_2 = (x_2, y_2, 1)$). Les quatre paramètres sont donc ces deux couples de coordonnées.

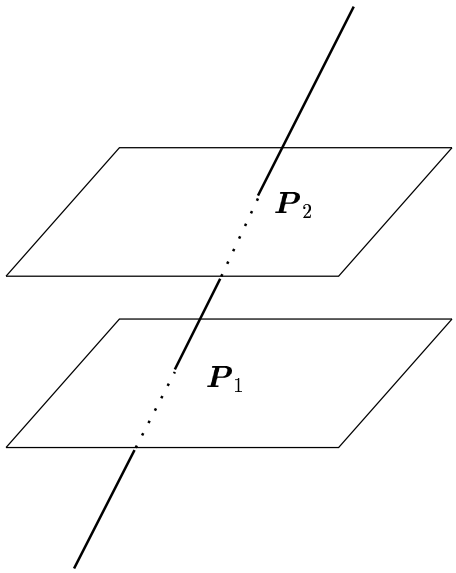


FIG. A.1: *Une droite possède quatre degrés de liberté.*

Toutefois, notre exemple montre le manque de généralité d'une représentation réduite, puisque certaines droites ne peuvent être représentées. On est alors amené à augmenter le nombre de paramètres, qui sont conséquemment liés.

A.1 Coordonnées de Plücker

Dans une note additionnelle à [Plü65], datée du 11 décembre 1865, Plücker a introduit huit représentations de droites 3D. Parmi elles, une seule a obtenue l'appellation de « coordonnées de Plücker ». Toutefois, nous préférons à cette appellation, celle, plus précise, de « coordonnées de Plücker projectives », car une autre de ces huit représentations peut être considérée comme la variante euclidienne de la précédente.

A.1.1 Coordonnées de Plücker projectives

Historiquement, Plücker a obtenu cette représentation en raisonnant sur les intersections des plans et droites puis par application de la dualité plan-point en dimension 3. Depuis, une présentation plus simple a été adoptée, à quelques variantes près [Fau93, GO97, Hun78, SK52] sur laquelle nous allons nous appuyer.

Considérons deux points quelconques, $\boldsymbol{P} = (x_1, x_2, x_3, x_4)$ et $\boldsymbol{P'} = (x'_1, x'_2, x'_3, x'_4)$, d'une même droite (D). Celle-ci est définie de manière unique par ces deux points. Cependant, cette représentation est loin d'être minimale puisqu'elle utilise huit paramètres pour fixer les quatre degrés de liberté de la droite. De plus, il n'y a pas de relation triviale entre ces paramètres.

Pour réduire le nombre de paramètres, on considère alors les mineurs 2×2 de la matrice $\left(\begin{smallmatrix} x_1 & x_2 & x_3 & x_4 \\ x'_1 & x'_2 & x'_3 & x'_4 \end{smallmatrix}\right)$. Six d'entre eux peuvent être indépendants. De plus, on vérifie aisément que ces mineurs sont uniques, à un facteur d'échelle près, pour tout choix des deux points de la droite en considérant les combinaisons linéaires de $\boldsymbol{P}$ et $\boldsymbol{P'}$ [Fau93, p24].

Les coordonnées de Plücker projectives sont alors définies par les six mineurs suivants :

$$\mathsf{D} = \begin{pmatrix} p_{41} & p_{42} & p_{43} & p_{23} & p_{31} & p_{12} \end{pmatrix}^T \quad avec \quad p_{ij} = x_i x'_j - x'_i x_j$$

La raison de ce choix apparaîtra dans le paragraphe A.1.2 pour son interprétation euclidienne.

Ces six coordonnées homogènes définissent donc un point, parfois appelé *point de Plücker* [GO97], de $\mathcal{P}^5$, l'espace projectif de dimension 5. Cela veut dire que seuls 5 coefficients sont effectivement indépendants. Il manque encore une relation pour se ramener aux quatre degrés de liberté d'une droite. Pour la trouver, il suffit d'exprimer le fait qu'une droite s'intersecte elle-même.

Soit donc deux droites (D_1) et (D_2) définies respectivement par les paires de points $(\boldsymbol{P}, \boldsymbol{P}')$ et $(\boldsymbol{Q}, \boldsymbol{Q}')$. Ces deux droites s'intersectent si ces quatre points sont coplanaires, soit

$$\begin{vmatrix} x_1 & x_2 & x_3 & x_4 \\ x'_1 & x'_2 & x'_3 & x'_4 \\ y_1 & y_2 & y_3 & y_4 \\ y'_1 & y'_2 & y'_3 & y'_4 \end{vmatrix} = 0$$

En développant ce déterminant, appelé *Grassmannien*, on aboutit à :

$$p_{41}q_{23} + p_{42}q_{31} + p_{43}q_{12} + p_{23}q_{41} + p_{31}q_{42} + p_{12}q_{43} = 0 \tag{A.1}$$

On obtient alors la relation manquante qui lie les coordonnées de Plücker entre elles :

$$p_{41}p_{23} + p_{42}p_{31} + p_{43}p_{12} = 0 \tag{A.2}$$

Cette équation définit un sous-espace de dimension 4 de $\mathcal{P}^5$, appelé *hypersurface de Plücker* [CEG+96] ou encore *quadrique de Klein* [PPR98]. Ainsi, tout point de cette quadrique correspond à une droite physique et vice-versa.

A.1.2 Coordonnées de Plücker euclidiennes

On considère parfois [BR79, Mac90] le point de Plücker D, associé à une droite (D), comme une paire de vecteurs (ou *bivecteur* [Mac90]) de la forme $\mathsf{D} = \left(\begin{smallmatrix}\mathbf{u}\\ \mathbf{v}\end{smallmatrix}\right)$ avec $\mathbf{u} = (p_{41}, p_{42}, p_{43})^T$ et $\mathbf{v} = (p_{23}, p_{31}, p_{12})^T$.

Alors, les relations (A.1) et (A.2) se réécrivent :

$$\mathbf{u}_1^T\mathbf{v}_2 + \mathbf{u}_2^T\mathbf{v}_1 = 0 \tag{A.3}$$

et

$$\mathbf{u}^T\mathbf{v} = 0 \tag{A.4}$$

Pour passer du projectif à l'euclidien, il suffit de fixer le facteur d'échelle. Ainsi, un point de coordonnées projectives (x, y, z, t), possède les coordonnées euclidiennes $(x/t, y/t, z/t, 1)$ (ou $(x, y, z, 0)$ s'il se trouve sur le plan à l'infini). De la même manière, on peut obtenir les coordonnées de Plücker euclidiennes à partir des coordonnées de Plücker projectives en fixant le facteur d'échelle. Pour cela, plutôt que de fixer une coordonnée, on impose, en reprenant la notion de bivecteur, une contrainte sur la norme du vecteur $\mathbf{u}$. Par conséquent, une droite, hors du plan à l'infini, est définie de manière unique par :

$$\mathsf{D} = \begin{pmatrix}\mathbf{u}\\ \mathbf{h}\end{pmatrix} \tag{A.5}$$

et la contrainte équivalente à (A.4) :

$$\underline{\mathbf{u}}^T \mathbf{h} = 0 \tag{A.6}$$

avec $\underline{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|}$ et $\mathbf{h} = \frac{\mathbf{v}}{\|\mathbf{u}\|}$. Si la droite est contenue dans le plan à l'infini, elle est alors définie par $\mathsf{D} = \left(\begin{smallmatrix}\mathbf{0}\\ \mathbf{v}\end{smallmatrix}\right)$.

Dans le cas euclidien, les coordonnées d'un point sont notés $(x, y, z, 1)$. En remplaçant les x_i par ces notations, on obtient :

$$\mathsf{D} = \begin{pmatrix} x'-x & y'-y & z'-z & yz'-z'y & zx'-xz' & xy'-y'z \end{pmatrix}^T$$

Le vecteur $\mathbf{u}$ n'est alors que le vecteur $\overrightarrow{\boldsymbol{PP'}}$ et le vecteur $\mathbf{v}$ est le produit vectoriel $\boldsymbol{P}\times\overrightarrow{\boldsymbol{PP'}}$. Par conséquent, $\underline{\mathbf{u}}$ est le vecteur directeur unitaire de la droite et $\mathbf{h} = \boldsymbol{P}\times\underline{\mathbf{u}}$. Ceci explique le choix des coordonnées de Plücker projectives.

A.1.3 Configurations de droites

Une des utilisations majeures des coordonnées de Plücker est la recherche d'ensembles de droites particulières (congruences, complexes, surfaces réglées et autres) qui apparaissent lors de la résolution de nombreux problèmes en robotique et en vision :

- étude de la mobilité des robots parallèles [Dan84, Mer88];
- cinématique [BR79, Hun78, LH98] ;
- ou encore, calcul de pose et reconstruction euclidienne à partir de droites [Nav93].

Le Théorème 31, en page 329, de [VY10] nous donne une définition de quelques-uns de ces ensembles :

> « *The lines whose coordinates satisfy one linear equation*
>
> $$a_{12}p_{12} + a_{13}p_{13} + a_{14}p_{14} + a_{34}p_{34} + a_{42}p_{42} + a_{23}p_{23} = 0$$
>
> *form a linear complex. Those whose coordinates satisfy two independent linear equations form a linear congruence, and those satisfying three independent linear equations form a regulus. Four independent linear equations are satisfied by two (distinct or coincident) lines, which may be improper* »

Ces ensembles correspondent donc à des sous-variétés de l'espace projectif $\mathcal{P}^5$, dans lesquelles on rencontre aussi bien des droites physiques qu'*imaginaires*, c.-à-d. qui n'appartiennent pas à la quadrique de Klein. Ces sous-variétés ont été étudiées par H. Grassmann (1807–1877) — dont les travaux n'ont été que très récemment traduits en français [Gra94]. Elles ont été depuis entièrement caractérisées [VY10] et illustrées [Dan84, Mer88].

Nous nous limiterons ici à donner l'exemple classique du regulus. Un regulus est défini comme étant l'ensemble des droites physiques intersectant 3 droites fixes et non sécantes. Il est ainsi l'intersection d'une sous-variété de dimension 3, puisqu'il vérifie 3 contraintes, et de la quadrique de Klein. Il forme donc un hyperboloïde à une nappe (Figure A.2) qui, soit dit en passant, permet de construire des aéro-réfrigérents[1] à l'aide de longerons rectilignes.

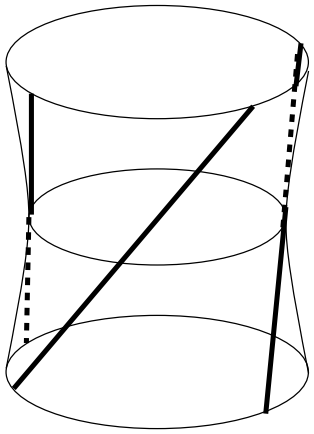


FIG. A.2: *Un hyperboloïde à une nappe est l'ensemble des droites intersectant trois droites fixes.*

A.2 Lien avec la cinématique

Une des huit représentations de Plücker lui permet d'entamer une «*geometry of forces*». En effet, il identifie les trois premières coordonnées projectives, le vecteur que nous avons noté $\mathbf{u}$, à une force et les trois dernières, notre vecteur $\mathbf{v}$, au moment de cette force.

Similairement, on peut assimiler un torseur cinématique $\binom{\mathbf{V}}{\mathbf{\Omega}}$ à une droite $\binom{\mathbf{u}}{\mathbf{v}}$ en posant $\mathbf{u} = \mathbf{\Omega}$ et $\mathbf{v} = (\mathbf{I}_3 - \underline{\mathbf{u}}\underline{\mathbf{u}}^T)\mathbf{V}$. La relation (A.4) est ainsi vérifiée et $\binom{\mathbf{u}}{\mathbf{v}}$ représente donc bien une droite.

1. En clair, les cheminées des centrales nucléaires ...

A.3 Projection d'une droite dans l'image

Soit (D), une droite de coordonnées de Plücker projectives $(\mathbf{u}, \mathbf{h})$ et sa projection dans l'image (d). Les coefficients de l'équation de (d) sont contenus dans le vecteur d.

On peut alors écrire, à la suite de Faugeras et Mourrain [FM95] l'équation de projection :

$$\underline{\mathbf{h}} \approx \tilde{\mathbf{P}} \begin{pmatrix} \mathbf{u} \\ \mathbf{h} \end{pmatrix}$$

où $\approx$ dénote l'égalité projective, c.-à-d. à un facteur proportionnel non nul près. Quant à la matrice $\tilde{\mathbf{P}}$, elle est l'extension aux droites de la matrice de projection des points. Elle est introduite dans le cas projectif grâce à l'algèbre de Grassmann-Cayley. Dans le cas euclidien, elle se décompose en les blocs suivants :

$$\tilde{\mathbf{P}} = \mathbf{K}^{-T} \begin{pmatrix} \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{0}_{3\times 3} \\ As(\mathbf{t})\mathbf{R} & \mathbf{R} \end{pmatrix}$$

où, traditionnellement, $\mathbf{K}$ contient les paramètres intrinsèques de la caméra et $(\mathbf{R}, \mathbf{t})$ est la transformation entre le repère dans lequel est exprimée la droite et le repère caméra.

Ce produit se décompose donc comme pour le cas des points et de droite à gauche :

- Changement de repère 3D [BR79, p. 532] : ici, passage du repère dans lequel est exprimé la droite au repère caméra ;
- Projection canonique ;
- Passage du repère caméra au repère image.

Par conséquent, les coefficients de l'équation de (d) sont, aux changements de repère près, homogènes au vecteur $\mathbf{h}$:

$$\mathsf{d} \approx \mathbf{h}$$

Annexe B

Étalonnage laser-caméra

B.1 Étalonnage laser-caméra —

La première étape consiste à prendre plusieurs images de la mire que nous utilisons pour calibrer une caméra. Ces images doivent être prises de manière à ce que le laser couplé à la caméra crée un point sur la mire (Figure B.1). En pratique, on pourra soit bouger la mire, soit bouger la caméra pour obtenir ces différentes images.

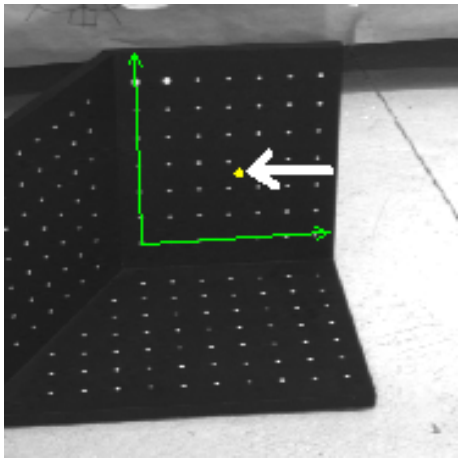


FIG. B.1: *La mire utilisée pour l'étalonnage de la caméra et un système d'axes associé au plan où se projette le faisceau laser (indiqué par la flèche).*

Chaque image de la mire permet d'étalonner la caméra et ainsi connaître la position/orientation de cette dernière par rapport à la mire ainsi que le modèle de la

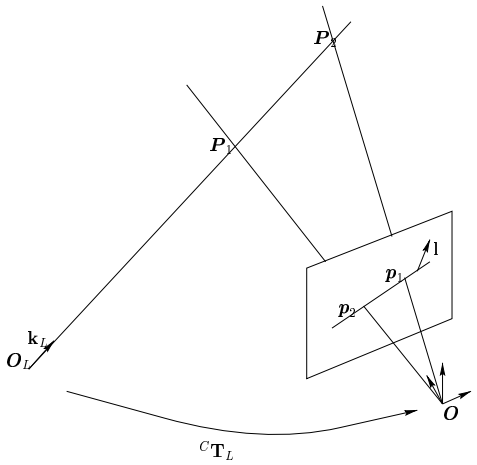


FIG. B.2: *Le mouvement de la mire devant le système caméra–laser fixe crée deux points* $\boldsymbol{P}_1$ *et* $\boldsymbol{P}_2$ *le long du faisceau laser. Ces points* 3D *se projettent en* $\boldsymbol{p}_1$ *et* $\boldsymbol{p}_2$. *Tous les points de la figure sont situés dans le plan perpendiculaire à* $\mathbf{l}$.

mire dans le repère caméra. Chaque image i permet aussi de déterminer dans quel plan Π_i de la mire se trouve le point laser (par exemple, en cherchant le point de la mire le plus proche du point laser). On peut alors en déduire la position 3D du point laser qui est l'intersection de la droite de vue passant par sa projection dans l'image et du plan Π_i dont on connaît désormais l'équation dans le repère caméra.

Considérons à présent le cas générique où nous avons bougé la mire devant la caméra (Figure B.2). Alors, nous disposons de 2 points image ($\boldsymbol{p}_1$ et $\boldsymbol{p}_2$) sur la droite épipolaire laser et de leurs coordonnées 3D dans le repère caméra ($\boldsymbol{P}_1$ et $\boldsymbol{P}_2$). Notons $\boldsymbol{O}_L$, l'origine du faisceau laser et $\mathbf{k}_L$ son vecteur directeur, exprimés dans le repère caméra ($\boldsymbol{O}xyz$).

Les points $\boldsymbol{p}_1$ et $\boldsymbol{p}_2$ permettent de définir l'équation de la droite épipolaire laser, de la forme : $ax + by + c = 0$. Cela définit aussi un vecteur normal à cette droite, $\mathbf{l} = (a, b, c)^T$, qui est le vecteur normal au plan d'interprétation passant par $\boldsymbol{p}_1$ et $\boldsymbol{p}_2$, mais aussi $\boldsymbol{P}_1$, $\boldsymbol{P}_2$ et $\boldsymbol{O}_L$. Ce plan contient aussi $\mathbf{k}_L$.

Par conséquent, on peut exprimer la condition d'appartenance de $\boldsymbol{O}_L$ au plan d'interprétation par :

$$\boldsymbol{O}_L^T \mathbf{l} = 0 \tag{B.1}$$

et l'orthogonalité de $\mathbf{l}$ avec ce plan, en confondant le point $\boldsymbol{O}_L$ et le vecteur $\overrightarrow{\boldsymbol{O}\boldsymbol{O}_L}$, par :

$$\mathbf{l} = s\, \boldsymbol{O}_L \times \mathbf{k}_L. \tag{B.2}$$

On remarquera que le point $\boldsymbol{O}_L$ peut être placé arbitrairement en n'importe quel position sur le faisceau. Nous le choisirons par conséquent dans le plan $z = 0$

de la caméra par souci de simplicité, ce qui est toujours possible en dehors du cas dégénéré où le faisceau laser est parallèle au plan image. Si l'on note $(x_L, y_L, 0)^T$, les coordonnées de $\boldsymbol{O}_L$, l'équation (B.1) devient :

$$ax_L + by_L = 0,$$

ce qui implique que $\boldsymbol{O}_L = s'\mathbf{t}_L$ où $\mathbf{t}_L = (-b, a, 0)^T$. En reportant ce résultat dans (B.2), on obtient :

$$\mathbf{l} = ss'\,\mathbf{t}_L \times \mathbf{k}_L$$

On peut donc exprimer $\mathbf{k}_L$ comme une combinaison linéaire de $\mathbf{t}_L \times \mathbf{l}$ et de $\mathbf{t}_L$. En considérant que $\mathbf{k}_L$ est normé, cette combinaison devient :

$$\mathbf{k}_L = \cos\theta \underbrace{\frac{\mathbf{t}_L \times \mathbf{l}}{\|\mathbf{t}_L \times \mathbf{l}\|}}_{\mathbf{k}'} + \sin\theta \underbrace{\frac{\mathbf{t}_L}{\|\mathbf{t}_L\|}}_{\mathbf{k}''} \tag{B.3}$$

Remarque : On peut aboutir à ce résultat en définissant la transformation rigide ${}^C\mathbf{T}_L = \left(\begin{smallmatrix} \mathbf{R}_L & \boldsymbol{O}_L \\ 0 & 1 \end{smallmatrix}\right)$ faisant passer du repère laser au repère caméra, puis en considérant que le faisceau laser est l'axe optique d'une caméra virtuelle et en appliquant la contrainte épipolaire.

Avec ces notations, les points $\boldsymbol{P}_i$, $i = 1, 2$, appartenant au faisceau laser, s'écrivent :

$$\boldsymbol{P}_i = \boldsymbol{O}_L + \alpha_i \mathbf{k}_L, \;\; i = 1, 2$$

ou, en faisant apparaître les inconnues (s', α_i, θ) :

$$\boldsymbol{P}_i = s'\mathbf{t}_L + \alpha_i(\cos\theta\mathbf{k}' + \sin\theta\mathbf{k}''), \;\; i = 1, 2.$$

Ce système d'équations non-linéaires donne aisément lieu à une solution linéaire. En effet, le système non-linéaire engendre le système linéaire suivant :

$$\underbrace{\begin{pmatrix} \mathbf{t}_L & \mathbf{k}' & \mathbf{k}'' & 0 & 0 \\ \mathbf{t}_L & 0 & 0 & \mathbf{k}' & \mathbf{k}'' \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} s' \\ \alpha_1 \cos\theta \\ \alpha_1 \sin\theta \\ \alpha_2 \cos\theta \\ \alpha_2 \sin\theta \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \end{pmatrix} \tag{B.4}$$

qui est de dimension (6×5) et de rang 4 (car $\mathbf{t}_L = \|\mathbf{t}_L\|\mathbf{k}''$). Il admet donc une infinité de solutions de la forme :

$$\begin{pmatrix} s' \\ \alpha_1 \cos\theta \\ \alpha_1 \sin\theta \\ \alpha_2 \cos\theta \\ \alpha_2 \sin\theta \end{pmatrix} = \underbrace{\mathbf{A}^\dagger \begin{pmatrix} \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \end{pmatrix}}_{\mathbf{v}} + \lambda\mathbf{w} \tag{B.5}$$

où $\mathbf{w}$ est un vecteur du noyau de $\mathbf{A}$ et † dénote l'inverse généralisée basée sur une décomposition SVD [PTVF92, §2.6], par exemple, $\mathbf{w} = (1, 0, -\|\mathbf{t}_L\|, 0, -\|\mathbf{t}_L\|)^T$.

Le facteur λ reste à déterminer. Pour cela, on utilise la contrainte sur les inconnues :

$$(\alpha_1 \cos\theta)(\alpha_2 \sin\theta) - (\alpha_2 \cos\theta)(\alpha_1 \sin\theta) = 0$$

d'où, en utilisant la notation v_i pour la i^{e} coordonnée de $\mathbf{v}$:

$$(v_1 + \lambda w_1)(v_4 + \lambda w_4) - (v_2 + \lambda w_2)(v_3 + \lambda w_3) = 0. \tag{B.6}$$

qui se simplifie grâce à l'expression algébrique de $\mathbf{w}$ en un polynôme de degré 1 en la seule inconnue λ, qui a toujours une solution unique en dehors du cas dégénéré où le faisceau laser est parallèle au plan image.

En résumé, l'algorithme d'étalonnage proposé est donc :

- Prendre plusieurs images de la mire de manière à ce que le faisceau laser crée un point sur la mire
- Calibrer la caméra et en déduire les coordonnées 3D des différents points laser
- Calculer $\mathbf{l}$ avec les données image et en déduire $\mathbf{t}_L$
- Résoudre le système linéaire (B.4) grâce aux équations (B.5) et (B.6) pour obtenir $\mathbf{k}_L$ et $\boldsymbol{O}_L$.

NB : En pratique, on s'assurera que la distance entre la mire et la caméra varie beaucoup afin d'obtenir une disparité importante le long de la droite épipolaire laser et une bonne précision sur le calcul de $\mathbf{k}_L$ et $\boldsymbol{O}_L$.

B.2 Triangulation laser-caméra

Lorsque le faisceau laser est étalonné, c.-à-d. que l'on connaît $\mathbf{k}_L$ et $\boldsymbol{O}_L$, il ne reste plus qu'à faire de la triangulation pour obtenir la position 3D $\boldsymbol{P}$ du point laser dans le repère caméra. Pour ce faire, nous exprimons que $\boldsymbol{P}$ appartient simultanément au faisceau laser et à la droite de vue passant par sa projection p :

$$\begin{aligned} \boldsymbol{P} &= \boldsymbol{O}_L + \alpha \mathbf{k}_L \\ \boldsymbol{P} &= \beta \boldsymbol{p} \end{aligned}$$

En soustrayant ces deux équations, on obtient un système linéaire à 2 inconnues (α et β) et 3 équations, qui nous donne la solution.

Connaissant à présent $\boldsymbol{P}$, on peut aisément retrouver le facteur d'échelle manquant au calcul de pose. Rappelons que nous avons été capable de calculer l'orientation $\mathbf{R}$ et la translation à un facteur d'échelle près $\lambda\mathbf{t}$ de la pièce de bateau par rapport à la caméra. Lorsque l'on connaît $\boldsymbol{P}_p$, la position du point laser dans le repère de la pièce, il est trivial de retrouver λ puisque $\boldsymbol{P} = \mathbf{R}\boldsymbol{P}_p + \lambda\mathbf{t}$.

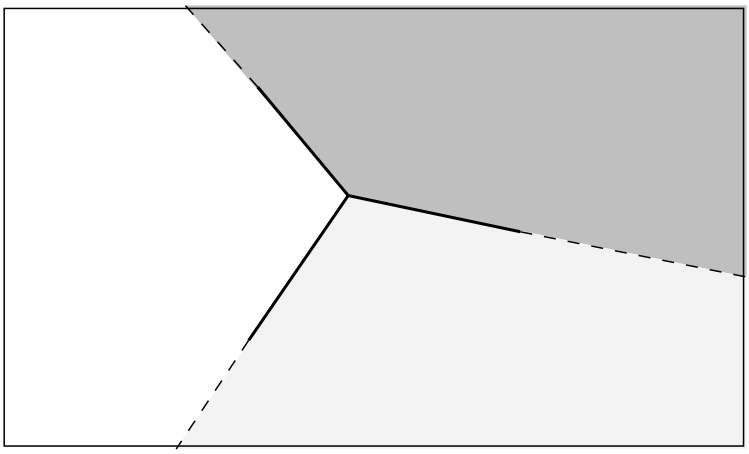


FIG. B.3: *Les trois segments de droites déterminent une partition de l'image correspondant à la projection des 3 plans du trièdre.*

Pour connaître $\boldsymbol{P}_p$, on va détecter dans l'image à quel plan de la pièce il appartient. Cela est assez simple puisque les arêtes de la pièce génèrent des demi-droites dans l'image et donc trois secteurs (Figure B.3). Alors, on a une base de ce plan $(\boldsymbol{O}_p, \mathbf{u}, \mathbf{v})$, où $\boldsymbol{O}_p$ est le centre du trièdre, $\mathbf{u}$ est le vecteur directeur d'une des deux arêtes du plan et $\mathbf{v}$, l'autre. Ces deux vecteurs directeurs sont obtenus directement à partir du modèle CAO. Le point $\boldsymbol{P}_p$ s'écrit alors :

$$\boldsymbol{P}_p = \boldsymbol{O}_p + s_u\mathbf{u} + s_v\mathbf{v}$$

ce qui donne dans le repère caméra :

$$\boldsymbol{P} = s_u\mathbf{R}\mathbf{u} + s_v\mathbf{R}\mathbf{v} + \lambda\mathbf{t},$$

système linéaire non homogène de rang plein, dont la solution unique nous fournit λ. Le système est effectivement de rang plein car sinon on aurait une configuration dans laquelle l'axe optique appartient au plan où se projette le point laser. Cela voudrait dire que ce plan est vu par la tranche. De plus, le point laser ne pourrait alors pas être détecté.

Annexe C

Compléments à l'étalonnage pince-caméra

C.1 Preuve du Théorème 7

Énonçons d'abord quelques résultats intermédiaires.

Proposition 5

Soient deux matrices de rotations, $\mathbf{R}$ *et* $\mathbf{R}'$*, conjuguées, c.-à-d. telles qu'il existe une matrice de rotation* $\mathbf{R}_X$ *telle que* $\mathbf{R}' = \mathbf{R}_X\mathbf{R}\mathbf{R}_X^T$*. Alors,*

1) si $\mathbf{v}$ *est un vecteur propre de* $\mathbf{R} \otimes \mathbf{R}'$*, alors* $(\mathbf{I} \otimes \mathbf{R}_X^T)\mathbf{v}$ *est un vecteur propre de* $\mathbf{R} \otimes \mathbf{R}$ *pour la même valeur propre ;*

2) si $\mathbf{x}$ *est un vecteur propre de* $\mathbf{R} \otimes \mathbf{R}$*, alors* $(\mathbf{I} \otimes \mathbf{R}_X)\mathbf{x}$ *est un vecteur propre de* $\mathbf{R} \otimes \mathbf{R}'$ *pour la même valeur propre.*

Preuve :

1) Soit $\mathbf{v}$ un vecteur propre de $\mathbf{R}\otimes\mathbf{R}'$ associé à la valeur propre λ. Alors, $(\mathbf{R}\otimes\mathbf{R}')\mathbf{v} = \lambda\mathbf{v}$. En remplaçant $\mathbf{R}'$ par $\mathbf{R}_X\mathbf{R}\mathbf{R}_X^T$ dans cette relation nous fournit :

$$(\mathbf{R} \otimes (\mathbf{R}_X\mathbf{R}\mathbf{R}_X^T))\mathbf{v} = \lambda\mathbf{v}$$

De la relation $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$[Bel60], nous tirons :

$$(\mathbf{I} \otimes \mathbf{R}_X)(\mathbf{R} \otimes \mathbf{R})(\mathbf{I} \otimes \mathbf{R}_X^T)\mathbf{v} = \lambda\mathbf{v}$$

Comme $(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$[Bel60], nous obtenons la relation suivante :

$$(\mathbf{I} \otimes \mathbf{R}_X^T)^{-1}(\mathbf{R} \otimes \mathbf{R})(\mathbf{I} \otimes \mathbf{R}_X^T)\mathbf{v} = \lambda \mathbf{v}$$

D'où, $(\mathbf{R} \otimes \mathbf{R})(\mathbf{I} \otimes \mathbf{R}_X^T)\mathbf{v} = \lambda(\mathbf{I} \otimes \mathbf{R}_X^T)\mathbf{v}$.

2) Soit $\mathbf{x}$, un vecteur propre de $\mathbf{R} \otimes \mathbf{R}$ associé à la valeur propre α. Alors,

$$\begin{aligned}
(\mathbf{R} \otimes \mathbf{R})\mathbf{x} &= \alpha \mathbf{x} \\
(\mathbf{R} \otimes \mathbf{R})(\mathbf{I} \otimes \mathbf{R}_X^T)(\mathbf{I} \otimes \mathbf{R}_X)\mathbf{x} &= \alpha(\mathbf{I} \otimes \mathbf{R}_X^T)(\mathbf{I} \otimes \mathbf{R}_X)\mathbf{x} \\
(\mathbf{I} \otimes \mathbf{R}_X)(\mathbf{R} \otimes \mathbf{R})(\mathbf{I} \otimes \mathbf{R}_X^T)(\mathbf{I} \otimes \mathbf{R}_X)\mathbf{x} &= \alpha(\mathbf{I} \otimes \mathbf{R}_X)\mathbf{x} \\
(\mathbf{R} \otimes \mathbf{R}')(\mathbf{I} \otimes \mathbf{R}_X)\mathbf{x} &= \alpha(\mathbf{I} \otimes \mathbf{R}_X)\mathbf{x}.
\end{aligned}$$

□

Proposition 6

Soient $\mathbf{R}_1$ et $\mathbf{R}_2$ deux matrices de rotation d'axes non parallèles. Soit $\mathbf{R}$ une autre matrice de rotation. Alors,

$$\left.\begin{array}{l} \mathbf{R}_1 \otimes \mathbf{R}_1 \; vec(\mathbf{R}) = vec(\mathbf{R}) \\ \mathbf{R}_2 \otimes \mathbf{R}_2 \; vec(\mathbf{R}) = vec(\mathbf{R}) \end{array}\right\} \Rightarrow \mathbf{R} = \mathbf{I}_3$$

Preuve :

Le système ci-dessus est équivalent à :

$$\begin{aligned} \mathbf{R}_1\mathbf{R} &= \mathbf{R}\mathbf{R}_1 \\ \mathbf{R}_2\mathbf{R} &= \mathbf{R}\mathbf{R}_2 \end{aligned}$$

Si $\mathbf{R}$ satisfait la première équation, alors, soit $\mathbf{R}$ est la matrice identité, soit elle a le même axe de rotation que $\mathbf{R}_1$. Similairement, soit elle est la matrice identité, soit elle a le même axe de rotation que $\mathbf{R}_2$. Comme $\mathbf{R}_1$ et $\mathbf{R}_2$ ont des axes de rotation différents, $\mathbf{R}$ est forcément l'identité. □

Proposition 7

Soient $\mathbf{R}_1$ et $\mathbf{R}_2$ deux matrices de rotation d'axes non parallèles. Soit $\mathbf{M}$ une matrice telle que :

$$\begin{aligned} \mathbf{R}_1 \otimes \mathbf{R}_1 \; vec(\mathbf{M}) &= vec(\mathbf{M}) \\ \mathbf{R}_2 \otimes \mathbf{R}_2 \; vec(\mathbf{M}) &= vec(\mathbf{M}) \end{aligned}$$

Alors,

$$\Rightarrow \exists \lambda \neq 0, \mathbf{M} = \lambda \mathbf{I}_3$$

Preuve :

Écrire

$$\mathbf{R}_1 \otimes \mathbf{R}_1 \; vec(\mathbf{M}) = vec(\mathbf{M})$$

est équivalent à dire que $\mathbf{R}_1$ et $\mathbf{M}$ commutent. Par conséquent, $\mathbf{M}$ est de la forme $\lambda\mathbf{R}$ où $\lambda \neq 0$ et $\mathbf{R}$ est une matrice de rotation qui commute avec $\mathbf{R}_1$. On peut le voir aisément en remplaçant $\mathbf{M}$ par sa décomposition en valeurs singulières.

Ainsi, $\mathbf{M} = \lambda\mathbf{R}$ où $\mathbf{R}$ est telle que :

$$\mathbf{R}_1\mathbf{R} = \mathbf{R}\mathbf{R}_1$$
$$\mathbf{R}_2\mathbf{R} = \mathbf{R}\mathbf{R}_2$$

D'après la Proposition 6, nous obtenons $\mathbf{R} = \mathbf{I}_3$ et $\mathbf{M} = \lambda\mathbf{I}_3$. □

Preuve du Théorème 7 — Le système (1.11) est équivalent à :

$$\begin{aligned} \mathbf{R}_{A_1} \otimes \mathbf{R}_{B_1}\mathbf{v} &= \mathbf{v} \\ \mathbf{R}_{A_2} \otimes \mathbf{R}_{B_2}\mathbf{v} &= \mathbf{v} \end{aligned}$$

Sous l'hypothèse que les déplacements de la caméra et ceux du robot sont conjugués par la transformation pince-caméra constante $(\mathbf{R}_X, \mathbf{t}_X)$ et d'après la Proposition 5, ce système devient :

$$\begin{aligned} \mathbf{R}_{B_1} \otimes \mathbf{R}_{B_1}\mathbf{v}' &= \mathbf{v}' \\ \mathbf{R}_{B_2} \otimes \mathbf{R}_{B_2}\mathbf{v}' &= \mathbf{v}' \end{aligned}$$

où $\mathbf{v}' = (\mathbf{I} \otimes \mathbf{R}_X^T)\mathbf{v}$. En appliquant le résultat de la Proposition 7, nous avons donc $vec^{-1}(\mathbf{v}') = \lambda\mathbf{I}_3$. Il ne reste plus qu'à insérer la définition de $\mathbf{v}'$ et à utiliser les propriétés du produit de Kronecker pour obtenir :

$$vec^{-1}(\mathbf{v})\mathbf{R}_X^T = \lambda\mathbf{I}_3$$

d'où

$$\mathbf{V} = vec^{-1}(\mathbf{v}) = \lambda\mathbf{R}_X$$

La matrice $\mathbf{V}$ du noyau du système (1.11) est donc proportionnelle à la rotation cherchée. Il ne reste alors plus qu'à trouver le coefficient de proportionnalité que l'on obtient à partir de la contrainte $det(\mathbf{R}_X) = 1$:

$$det(\mathbf{V}) = \lambda^3$$

d'où

$$\lambda = sgn(det(\mathbf{V})) \; |det(\mathbf{V})|^{1/3}$$

□

C.2 Transformation "moyenne"

En effectuant n tirages aléatoires de k images parmi une séquence disponible, on peut calculer n estimations de la transformation pince-caméra. À partir de ces n estimations, on veut calculer une transformation pince-caméra "moyenne".

La définition du sens de "moyenne" est ardue car on ne peut pas utiliser la moyenne arithmétique de matrices de déplacement rigide pour obtenir une matrice de déplacement rigide. En effet, la somme de $n > 1$ matrices de rotation n'est pas une matrice de rotation. Si l'on utilise les quaternions unitaires pour représenter la rotation, et *a fortiori* les quaternions duaux unitaires pour représenter les déplacements rigides, le problème est le même : la moyenne arithmétique ne conserve pas les contraintes associées à chaque représentation.

On ira voir [Pen96] pour une étude mathématique poussée du problème. Nous préférons cependant nous contenter d'une approche plus simple, mais d'une efficacité comparable. Nous travaillerons donc sur des représentations réduites des rotations : angles Roulis-Tangage-Lacet ou représentation par axe et angle, par exemple.

C.2.1 Angles Roulis-Tangage-Lacet

L'avantage d'une telle représentation est qu'elle est minimale. En revanche, il y a des ambiguïtés qui apparaissent dans l'extraction de ces angles de la matrice de rotation. En dehors de ces ambiguïtés, aisément détectables, on obtient une représentation unique de la rotation. On peut alors calculer une moyenne sur les n triplets d'angles extraits des n estimations de la rotation pince-caméra et reconstruire la rotation "moyenne" à partir de ces angles moyens.

La non-unicité de la mesure de l'angle peut, si l'on n'y prend garde, biaiser le calcul. En effet, lorsque l'on considère la mesure principale des angles (c'est-à-dire prise sur $[0, 2\pi[$) et que l'angle réel que l'on estime est proche de 0, on peut avoir des estimations de cet angle proches de 2π (cf Figure C.1) qui vont tirer, à tort, la moyenne vers π. Un problème similaire interviendra lorsque l'angle réel estimé sera proche de π et que l'on exprimera les angles dans $]-\pi, \pi]$. On en déduit donc la méthode suivante pour calculer la moyenne d'angles.

Soient $\alpha \in [0, 2\pi[$, l'angle réel à estimer et α' son expression dans $]-\pi, \pi]$. Soient encore $\alpha_i \in [0, 2\pi[$, ses estimations et α'_i, l'expression des α_i dans $]-\pi, \pi]$.

1. Calculer les moyennes $\bar{\alpha} = \frac{1}{n}\sum_{i=1}^{n} \alpha_i$ et $\bar{\alpha'} = \frac{1}{n}\sum_{i=1}^{n} \alpha'_i$
2. Calculer les variances $\sigma_\alpha^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\alpha_i - \bar{\alpha})^2$ et $\sigma_{\alpha'}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\alpha'_i - \bar{\alpha'})^2$
3. Si $\sigma_\alpha > \sigma_{\alpha'}$, alors $\alpha' \approx \bar{\alpha'}$; sinon, $\alpha \approx \bar{\alpha}$
4. Ramener le résultat dans l'intervalle souhaité.

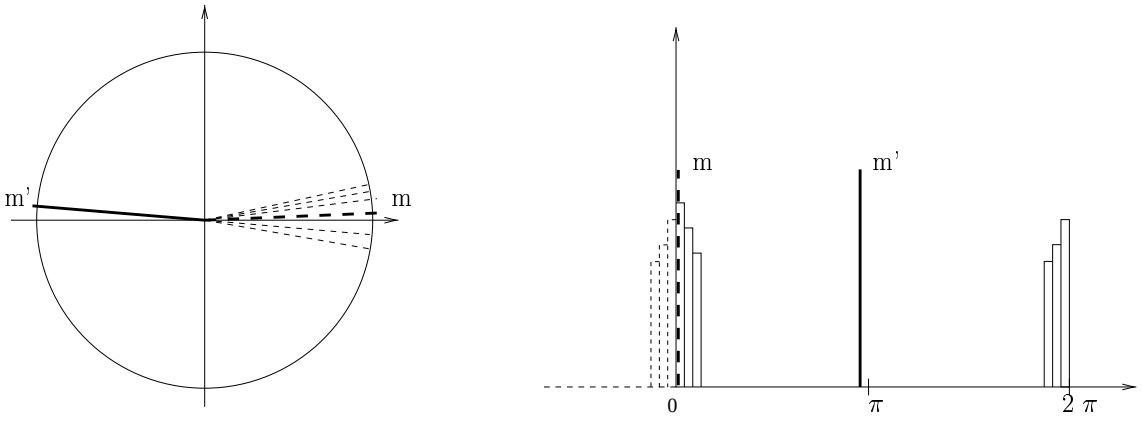


FIG. C.1: *La moyenne d'une distribution d'angles centrée sur 0 peut donner 2 résultats selon si l'on se place sur* $]-\pi,\pi]$ *(m) ou sur* $[0,2\pi[$ *(m').*

Pour retrouver la transformation "moyenne", on fait alors la moyenne des angles Roulis-Tangage-Lacet pour en déduire la rotation associée, puis on fait la moyenne des translations.

C.2.2 Axe/angle

Cette représentation n'est pas minimale mais presque. En effet, la contrainte qui impose que la norme de l'axe de rotation soit unitaire peut être supprimée en incorporant la valeur de l'angle dans l'axe : en multipliant le vecteur unitaire par l'angle, on obtient un vecteur de dimension 3 qui représente à lui seul la rotation.

De plus, cette contrainte n'est pas aussi gênante que celles intervenant dans les représentations matricielle et par quaternion. En effet, si la somme de vecteurs unitaires n'est pas un vecteur unitaire, il est cependant trivial de trouver le vecteur unitaire associé à cette somme.

Cette représentation est ambiguë car $(\theta,\mathbf{n})$ et $(-\theta,-\mathbf{n})$ sont deux couples axe/angle qui représentent la même rotation. Pour éviter cette ambiguïté, on choisit de prendre θ dans l'intervalle $[0,2\pi[$. Cependant, lorsque $\theta\approx\pi$, l'ambiguïté n'est pas levée par cette convention car les couples $(\pi,\mathbf{n})$ et $(\pi,-\mathbf{n})$ représentent la même rotation.

Lorsque l'on détecte que l'on est dans un voisinage de π (en calculant la moyenne des angles sur $[0,2\pi[$), on peut alors lever l'ambiguïté afférente en forçant un coefficient de l'axe de rotation à être positif. À cause de l'ambiguïté autour de π, il est difficile de savoir si l'on doit propager le changement de signe vers l'angle. On va donc choisir entre la moyenne des angles avec et sans propagation du signe selon la plus petite variance. On obtient alors la méthode suivante :

Soient $(\theta_i,\mathbf{n}_i)$, les couples axe/angles estimés avec $\theta_i\in[0,2\pi[$

1. Calculer $\bar{\theta}$, l'angle moyen de la rotation à partir des angles θ_i.

2. Calculer σ_θ, l'écart-type des angles θ_i par rapport à $\bar{\theta}$
3. Si $\bar{\theta} \approx \pi$, alors
   (a) Choisir un coefficient c de l'axe de rotation
   (b) Pour tout i
      - $\mathbf{n}'_i = sgn(\mathbf{n}_i^c)\mathbf{n}_i$
      - $\theta'_i = sgn(\mathbf{n}_i^c)\theta_i$
   (c) Calculer $\bar{\theta}'$
   (d) Calculer $\sigma_{\theta'}$, l'écart-type des angles θ'_i par rapport à $\bar{\theta}'$
   (e) Si $\sigma_\theta > \sigma_{\theta'}$, alors $\bar{\theta}'$ est le véritable angle moyen.
   (f) Calculer $\bar{\mathbf{n}'} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{n}'_i$
   (g) Normaliser $\bar{\mathbf{n}'}$

   sinon
   (a) Calculer $\bar{\mathbf{n}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{n}_i$
   (b) Normaliser $\bar{\mathbf{n}}$

C.2.3 Calcul robuste

On peut rendre plus robustes les deux méthodes précédentes en remplaçant la moyenne arithmétique par des estimateurs plus robustes. Le premier qui vient à l'esprit est l'opérateur médian, mais l'on peut aussi utiliser un M-estimateur, cf. [Rou97] pour une courte introduction et [Hub81, RL87, HRRS86] pour plus de détails.

Calcul du M-estimateur

On donne ici les deux méthodes de calcul du M-estimateur (présentées dans [Hub81], § 6.7 modifiées selon [HRRS86], § 2.3a).

On note, selon la tradition statistique, T, la moyenne et S, l'écart-type et x_i, les échantillons. Le principe est de remplacer le calcul de la moyenne $T = \frac{1}{n}\sum_{i=1}^{n} x_i$, très sensible aux mesures aberrantes par une estimation de celle-ci selon une méthode dérivée de l'estimation au maximum de vraisemblance (d'où l'appellation M-estimateur). On va donc minimiser[1]

$$\sum_{i=1}^{n} \rho(x_i - T) \tag{C.1}$$

1. On notera la similarité avec la méthode du maximum de vraisemblance qui minimise $\sum_{i=1}^{n} -\ln f(x_i, T)$.

où ρ est une fonction positive qui décroît lorsque $\|x_i - T\|$ croît, plutôt que $\sum (x_i - T)^2$, critère sous-jacent au calcul de la moyenne arithmétique.

La solution du problème de minimisation (C.1) peut être obtenue comme la solution de l'équation suivante, faisant intervenir la dérivée (ψ) de ρ :

$$\sum_{i=1}^{n} \psi(x_i - T) = 0 \tag{C.2}$$

qui, si l'on pose $w_i(x_i, T) = \frac{\psi(x_i - T)}{x_i - T}$, peut aussi s'écrire

$$\sum_{i=1}^{n} w_i(x_i, T) \cdot (x_i - T) = 0$$

On obtient ainsi une solution formelle, quoiqu'implicite, pour T sous la forme d'une moyenne pondérée :

$$T = \frac{\sum w_i(x_i, T) xi}{\sum w_i(x_i, T)} \tag{C.3}$$

On peut donc obtenir T en résolvant soit (C.2), soit (C.3). Dans le premier cas, on utilisera l'algorithme de Newton-Raphson [AW86] qui va réduire itérativement les résidus $x_i - T$ (méthode des résidus); dans le second, on va raffiner les poids $w_i(x_i, T)$ jusqu'à convergence (méthode des poids).

Dans les deux cas, il est nécessaire de normaliser les résidus par l'écart-type de l'échantillon (l'échelle, selon la terminologie statistique) car les M-estimateurs ne sont pas invariants par changement d'échelle. Cet écart-type dépend, bien entendu, de la moyenne et peut être mis à jour en même temps que celle-ci [Hub81]. Cependant, nous suivrons les recommandations de [HRRS86] et les estimerons séparément.

En pratique, nous avons utilisé la fonction $\psi(x) = \begin{cases} k\ sgn(x) & si\ |x| > k \\ x & si\ |x| \leq k \end{cases}$ avec $k = 1.345$ [Hub81].

Méthode des résidus

1. Calculer $T_0 = med(x_i)$
2. Calculer $S_0 = 1.483\ med(|x_i - T_0|)$
3. Itérer jusqu'à convergence :
$$T_{k+1} = T_k + \frac{\sum \psi\left[(x_i - T_k)/S_0\right] S_0}{\sum \psi'\left[(x_i - T_k)/S_0\right]}$$
4. $T = T_{k+1}$
5. $S = 1.483\ med(|x_i - T|)$

Méthode des poids

1. Calculer $T_0 = med(x_i)$
2. Calculer $S_0 = 1.483\ med(|x_i - T_0|)$
3. Itérer

$$T_{k+1} = T_k + \frac{\sum w_i^{(k)} x_i}{\sum w_i^{(k)}} \quad \text{avec} \quad w_i^{(k)} = \frac{\psi\left[(x_i - T_k)/S_0\right]}{(x_i - T_k)/S_0}$$

   jusqu'à convergence.

4. $T = T_{k+1}$
5. $S = 1.483\ med(|x_i - T|)$

Carte des vins

[And96] Wladimir Andreff. *Les multinationales globales.* La Découverte, Repères, 117, 1996. (p 3)

[ATYM93] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Trans. on Robotics and Automation*, 9(2):152–165, April 1993. (p 19)

[ATYM94] P. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Hand-Eye Coordination for Robotic Tracking and Grasping. In *Visual Servoing*, pages 33–70. K. Hashimoto, 1994. (p 19)

[AW86] K. Arbenz and A. Wohlhauser. Analyse numérique. In *Méthodes mathématiques pour l'ingénieur*. Presses polytechniques et universitaires romandes, 1986. (p 177)

[Bel60] R. Bellman. *Introduction to matrix analysis.* McGraw-Hill, 1960. (pp 109, 111, 171, 172)

[BJP93] Z. Bien, W. Jang, and J. Park. Characterization and use of feature-Jacobian matrix for visual servoing. In Hashimoto [Has93], pages 33–70. (p 17)

[Bou93] S. Boukir. *Reconstruction 3D d'un environnement statique par vision active.* Thèse de doctorat, Université de Rennes I, IRISA Rennes, France, October 1993. (p 31)

[BR79] O. Bottema and B. Roth. *Theoretical Kinematics.* Dover Publications, 1979. (pp 161, 162, 164)

[Bre78] J. W. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Trans. on Circuits and Systems*, CAS-25(9):772–781, 1978. (pp 63, 109)

[BS72] R. H. Bartels and G. H. Stewart. Solution of the equation AX+XB = C. *Communications of the ACM*, 15(9):820–826, 1972. (p 108)

[BSF88] Y. Bar-Shalom and Th. E. Fortmann. Tracking and data association. In *Mathematics in science and engineering*, volume 179. Academic Press, 1988. (pp 148, 149)

[CAD95] C. Colombo, B. Allota, and P. Dario. Affine Visual Servoing: A Framework for Relative Positionning with a Robot. In *Proc. IEEE International Conference on Robotics and Automation*, 1995. (pp 17, 19)

[CC96] G. M. T. Cross and R. Cipolla. Affine Visual Servoing. In *British Machine Vision Conference*, pages 425–434, 1996. (pp 17, 19)

[CC97] A. Crétual and F. Chaumette. Positioning a camera parallel to a plane using dynamic visual servoing. In *IEEE/RSJ/INRIA Workshop On New Trends in Image-based Robot Servoing*, pages 43–48, Grenoble, September 1997. (p 17)

[CEG+96] B. Chazelle, H. Edelsbrunner, L. J. Guibas, M. Sharir, and J. Stolfi. Lines in space : Combinatorics and algorithms. *Algorithmica*, 15:428–447, 1996. (p 161)

[CG96] P. I. Corke and M. C. Good. Dynamic effects in visual closed-loop systems. *IEEE Trans. on Robotics and Automation*, 12(5):671–683, 1996. (p 16)

[Cha90] F. Chaumette. *La relation vision-commande : théorie et application à des tâches robotques* . Thèse, Université de Rennes I, 1990. (pp 4, 16, 17, 21, 34, 37, 46)

[Cha97] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *Workshop on Vision and Control, Block Island, Rhode Island,*, June 1997. (pp 5, 16)

[Che91a] H. Chen. Pose determination from line-to-plane correspondences: Existence solutions and closed-form solutions. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 13(6):530–541, June 1991. (p 12)

[Che91b] H. H. Chen. A screw motion approach to uniqueness analysis of head-eye geometry. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 145–151, 1991. (pp 25, 116)

[Chr98] S. Christy. *Localisation et modélisation tridimensionnelles par approximations successives du modèle perspectif de caméra*. Thèse de doctorat, Institut National Polytechnique de Grenoble, GRAVIR – IMAG – INRIA Rhône–Alpes, 1998. (pp 12, 18, 143)

[CK91] J. C. K. Chou and M. Kamel. Finding the position and orientation of a sensor on a robot manipulator using quaternions. *International Journal of Robotics Research*, 10(3):240–254, June 1991. (p 24)

[Col99] C. Collewet. *Contributions à l'élargissement du champ applicatif des asservissements visuels 2D*. PhD thesis, Université de Rennes I, 1999. (pp 5, 17)

[Cor93] P. I. Corke. Visual control of robot manipulators — a review. In Hashimoto [Has93], pages 33–70. (p 15)

[Dan84] A. Dandurand. La rigidité des réseaux spatiaux composés. *Topologie structurale*, 10, 1984. (pp 162, 163)

[DBC96] K. Daniilidis and E. Bayro-Corrochano. The dual quaternion approach to hand-eye calibration. In *Proc. IAPR International Conference on Pattern Recognition*, pages 318–322, 1996. (pp 23, 119, 123, 125, 130, 133)

[DdAAB91] J. Dias, A. de Almeida, H. Araújo, and J. Batista. Improving camera calibration by using multiple frames in hand-eye robotic systems. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 285–291, 1991. (p 20)

[Deb96] C. Debain. *Lois de commande pour le contrôle et la mobilité des machines agricoles*. Thèse, Université Blaise Pascal (Clermont-Ferrand), 1996. (pp 17, 34)

[DFE93] V. De Filippos and R. Eleta. Les limites de la robotisation. *Problèmes économiques*, 2(350):12–18, 1993. (p 3)

[DH98] F. Dornaika and R. Horaud. Simultaneous robot-world and hand-eye calibration. IEEE *Transactions on Robotics and Automation*, 14(4), August 1998. (pp 20, 25)

[DN96] K. Deguchi and T. Noguchi. Visual servoing using eigenspace method and dynamic calculation of interaction matrices. In *Proc. IAPR International Conference on Pattern Recognition*, pages 302–309, 1996. (p 17)

[DP96] J. Domingo and J. Pelechano. Measurements and Storage of a Network of Jacobians as a Method for the Visual Positionning of a Robot Arm. *Journal of Intelligent and Robotic Systems*, 16:407–422, 1996. (pp 15, 18)

[DRLR89] M. Dhome, M. Richetin, J.T. Lapresté, and G. Rives. Determination of the attitude of 3D objects from a single perspective view. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989. (pp 12, 47, 74)

[DSH95] A. S. Deif, N. P. Seif, and S. A. Hussein. Sylvester's equation : accuracy and computational stability. *Journal of Computational and Applied Mathematics*, 61:1–11, 1995. (p 108)

[Dua96] G.-R. Duan. On the solution to the Sylvester matrix equation AV+BW = EVF. *IEEE Trans. on Automatic Control*, 41(4):612–314, 1996. (p 108)

[Eco97] Umberto Eco. *Comment voyager avec un saumon*. Grasset, 1997. (p 3)

[ECR92] B. Espiau, F. Chaumette, and P. Rives. A New Approach To Visual Servoing in Robotics. *IEEE Trans. on Robotics and Automation*, 8(3), June 1992. (pp 15, 16, 17, 21)

[Esp93] B. Espiau. Effect of Camera Calibration Errors on Visual Servoing in Robotics. In *Third International Symposium on Experimental Robotics*, October 1993. (p 15)

[Esp95] B. Espiau. Sur les erreurs en asservissement visuel. Rapport de Recherche 2619, INRIA, July 1995. (p 15)

[Fau93] O. Faugeras. *Three-Dimensional Computer Vision - A Geometric Viewpoint*. Artificial intelligence. The MIT Press, Cambridge, MA, USA, Cambridge, MA, 1993. (pp 13, 31, 32, 160)

[FLM91] J. T. Feddema, C. S. G. Lee, and O. R. Mitchell. Automatic selection of image features for visual servoing of a robot manipulator. *IEEE Trans. on Robotics and Automation*, 7, 1991. (p 17)

[FM95] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 951–956, June 1995. (p 164)

[FT87] O.D. Faugeras and G. Toscani. Camera calibration for 3D computer vision. In *Proceedings of International Workshop on Machine Vision and Machine Intelligence, Tokyo, Japan*, 1987. (p 10)

[GA93] M. S. Grewal and A. P. Andrews. *Kalman filtering : theory and practice.* Prentice–Hall, 1993. (p 142)

[Gir97] Georges Giralt. *La robotique.* Dominos. Flammarion, 1997. (p 3)

[GMOS95] E. Grosso, G. Metta, A. Oddera, and G. Sandini. Uncalibrated Visual Servoing in Reaching Tasks. Rapport de Recherche 3/95, LIRA-Lab - DIST University of Genova, April 1995. (pp 17, 19)

[GNL79] G. Golub, C. Nash, and C. Van Loan. A Hessenberg-Schur method for the problem AX+XB = C. *IEEE Trans. on Automatic Control*, 24:903–913, 1979. (p 108)

[GNTS96] V. Gengenbach, H.-H. Nagel, M. Tonko, and K. Schäfer. Automatic Dismantling Integrating Optical Flow into a Machine Vision-Controlled Robot System. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1320–1325, April 1996. (pp 5, 16)

[GO97] J. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry.* CRC Press, 1997. (p 160)

[Gra94] H. Grassmann. *La science de la grandeur extensive. La "Lineale Ausdehnungslehre".* Librairie Scientifique et Technique Albert Blanchard, 1994. Trad. et preface de Dominique Flament et Bernd Bekemeier. Trad. revue par Eberhard Knobloch. (p 163)

[GTX+96] B. K. Ghosh, T.-J. Tarn, N. Xi, Z. Yu, and Di Xiao. Calibration Free Visually Controlled Manipulation of Parts in a Robotic Manufacturing Workcell. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3197–3202, April 1996. (p 18)

[HA94] K. Hosoda and M. Asada. Versatile Visual Servoing without Knowledge of True Jacobian. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 186–193, 1994. (pp 18, 20)

[Hag97] G. D. Hager. A modular system for robust positioning using feedback from stereo vision. IEEE *Transactions on Robotics and Automation*, 13(4):582–595, August 1997. (pp 6, 17, 19, 34, 37, 43)

[Har94] R.I. Hartley. Projective reconstruction from line correspondences. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Seattle, Washington, USA*, pages 903–907, 1994. (p 13)

[Has93] K. Hashimoto, editor. *Visual Servoing — Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific, 1993. (pp 179, 181, 190)

[HC94] N. Hollinghurst and R. Cipolla. Uncalibrated Stereo Hand-Eye Coordination. *Image and Vision Computing*, 12(3):187–192, 1994. (p 19)

[HCDL95] R. Horaud, S. Christy, F. Dornaika, and B. Lamiroy. Object pose: Links between paraperspective and perspective. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 426–433, Cambridge, Mass., June 1995. IEEE Computer Society Press. (p 143)

[HCLL89] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics and Image Processing*, 47:33–44, 1989. (p 98)

[HCM95] G. D. Hager, W.-C. Chang, and A. S. Morse. Robot Hand-Eye Coordination based on Stereo Vision. *IEEE Control Syst. Mag.*, 15:30–39, February 1995. (pp 17, 19)

[HD95] R. Horaud and F. Dornaika. Hand-eye calibration. *International Journal of Robotics Research*, 14(3):195–210, June 1995. (pp 24, 119, 123, 125, 130, 133)

[HD97] G. D. Hager and Z. Dodds. A projective framework for constructing accurate hand-eye systems. In *IEEE/RSJ/INRIA Workshop On New Trends in Image-based Robot Servoing*, pages 71–82, 1997. (p 20)

[HDBM94] R. Horaud, F. Dornaika, B. Boufama, and R. Mohr. Self calibration of a stereo head mounted onto a robot arm. In J.O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden*, pages 455–462. Springer-Verlag, 1994. (p 24)

[HDE98] R. Horaud, F. Dornaika, and B. Espiau. Visually guided object grasping. IEEE *Transactions on Robotics and Automation*, 14(4), August 1998. (p 15)

[HDHM99] J. P. Hespanha, Z. Dodds, G. D. Hager, and A. S. Morse. What tasks can be performed with an uncalibrated stereo vision system? *International Journal on Computer Vision*, 1999. (pp 15, 20)

[HEK96] K. Hashimoto, T. Ebine, and H. Kimura. Visual Servoing with Hand-Eye Manipulator – Optimal Control Approach. In *Proc. IEEE International Conference on Robotics and Automation*, pages 766–774, October 1996. (p 15)

[HHC96] S. Hutchinson, G. D. Hager, and P. I. Corke. A Tutorial on Visual Servo Control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996. (p 15)

[Hor86] B. K. P. Horn. *Robot Vision*. MIT Press, 1986. (p 20)

[HR92] D. Y. Hu and L. Reichel. Krylov-subspace methods for the Sylvester equation. *Linear Algebra and its Applications*, 172:283–313, 1992. (p 108)

[HRRS86] F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel. *Robust Statistics : the Approach Based on Influence Functions*. Wiley series in probability and mathematical. John Wiley & Sons, Ltd., New York, 1986. (pp 176, 177)

[HSA95] K. Hosoda, K. Sakamoto, and M. Asada. Trajectory Generation for Obstacle Avoidance of Uncalibrated Stereo Visual Servoing without 3D Reconstruction. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 29–34, 1995. (pp 15, 20)

[Hub81] P.J. Huber. *Robust Statistics*, volume IX. John Wiley & Sons, Ltd., New York, 1981. (pp 176, 177)

[Hug59] V. Hugo. *La légende des siècles, second tome*. Gallimard/Flammarion, 1859. (p 34)

[Hun78] K.H. Hunt. *Kinematic Geometry of Mechanisms*. Clarendon Press, 1978. (pp 160, 162)

[JN95] M. Jägersand and R. Nelson. Visual Space Task Specification, Planning and Control. In *Proc. IEEE International Symposium on Computer Vision*, 1995. (pp 15, 18, 19)

[JS96] R. Joshi and A. C. Sanderson. Application of feature-based multi-view servoing for lamp filament alignment. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1306–1313, 1996. (p 19)

[JSW97] F. Janabi-Sharifi and W. J. Wilson. Automatic selection of images features for visual servoing. *IEEE Trans. on Robotics and Automation*, 13(6):890–903, 1997. (p 17)

[KD94] K. Kinoshita and K. Deguchi. Simultaneous Determination of Camera Pose and Intrinsic Parameters by Visual Servoing. In *Proc. IAPR International Conference on Pattern Recognition*, pages 285–289, October 1994. (p 15)

[Kha96] Hassan K. Khalil. *Nonlinear systems – 2nd ed.* Prentice-Hall, 1996. (p 63)

[Kim96a] Dongmin Kim. *Calibration problems in robotics*. PhD thesis, University of Michigan, 1996. (p 24)

[Kim96b] Dongmin Kim. Dual quaternion application to kinematic calibration of wrist-mounted camera. *Journal of Robotic Systems*, 13(3):153–162, 1996. (p 24)

[KRHK95] D. Kim, A. A. Rizzi, G. D. Hager, and D. E. Koditschek. A "Robust" Convergent Visual Servoing System. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 348–352, 1995. (p 15)

[LB87] M. Le Borgne. Quaternions et contrôle sur l'espace des rotations. Rapport 751, INRIA, 1987. (pp 50, 51)

[LB95] M. Li and D. Betsis. Head-eye calibration. In *International Conference on Computer Vision*, pages 40–45, 1995. (p 23)

[LG93] M. Lei and B. K. Ghosh. Visually guided robotic tracking and grasping of a moving object. In *Proceedings of the 32nd Conference on Decision and Control*, pages 1604–1609, 1993. (p 17)

[LH81] H.C. Longuet-Higgins. A computer program for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981. (p 17)

[LH86] Y. Liu and T.S. Huang. Estimation of rigid body motion using straight line correspondences, further results. In *Proceedings of the 8th International Conference on Pattern Recognition, Paris, France*, pages 306–307, October 1986. (p 13)

[LH98] J. Lenarčič and M. L. Husty, editors. *Advances in Robot Kinematics: Analysis and Control*. Kluwer Academic Publishers, 1998. (p 162)

[LHF90] Y. Liu, T.S. Huang, and O.D. Faugeras. Determination of camera location from 2D to 3D line and point. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, January 1990. (p 13)

[Li98] M. Li. Kinematic calibration of an active head-eye system. *IEEE Trans. on Robotics and Automation*, 14(1):153–158, February 1998. (p 113)

[LMH96] C-P Lu, E Mjolsness, and G D Hager. Online computation of exterior orientation with application to hand-eye calibration. *Mathematical and Computer Modelling*, 24(5–6):121–144, 1996. (p 20)

[Low85] D.G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, Massachusets, 1885. (p 12)

[Low87] D. G. Lowe. Three-dimensionnal object recognition from single two-dimensionnal images. *Artificial Intelligence*, 31:355–395, 1987. (p 146)

[LR96] S. Lee and S. Ro. A self-calibration model for hand-eye systems with motion estimation. *Mathl. Comput. Modelling*, 24(5/6):49–77, 1996. (p 24)

[LT88] R.K. Lenz and R.Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 10(5):713–720, September 1988. (p 10)

[Mac90] J. M. MacCarthy. *Introduction to Theoretical Kinematics*. MIT Press, 1990. (pp 35, 161)

[Mal98] E. Malis. *Contributions à la modélisation et à la commande en asservissement visuel*. PhD thesis, Université de Rennes I, 1998. (pp 17, 46)

[MBG96] P. Martinet, F. Berry, and J. Gallice. Use of first derivative of geometric features in Visual Servoing. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3413–3419, April 1996. (p 17)

[MCB99] E. Malis, F. Chaumette, and S. Boudet. 2-1/2-D visual servoing. *IEEE Trans. on Robotics and Automation*, 15(2):238–250, 1999. (p 17)

[MDGD97] P. Martinet, N. Daucher, J. Gallice, and M. Dhome. Robot control using monocular pose estimation. In *IEEE/RSJ/INRIA Workshop On New Trends in Image-based Robot Servoing*, pages 1–12, Grenoble, 1997. (pp 16, 47)

[Mer88] J.-P. Merlet. Parallel manipulators, part 2 : Theory. singular configurations and grassmann geometry. Rapport de Recherche 791, INRIA, 1988. (pp 162, 163)

[MKNM93] N. Maru, H. Kase, A. Nishikawa, and F. Miyazaki. Manipulator Control by Visual Servoing with the Stereo Vision. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1866–1870, 1993. (p 19)

[MLS94] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994. (p 50)

[MM91] F. Miyazaki and Y. Masutani. Robustness of sensory feedback control based on imperfect Jacobian. In *Proc. International Symposium on Robotics Research*, pages 201–208, August 1991. (p 15)

[Mot92] G. Motyl. *Couplage d'une caméra et d'un faisceau laser en commande référencée vision.* Thèse, Université Blaise Pascal (Clermont-Ferrand), 1992. (pp 17, 34)

[Nav93] N. Navab. *Visual motion of lines and cooperation between motion and stereo.* Thèse, Université Paris-Sud, 1993. (p 162)

[Neu69] H. Neudecker. A note on Kronecker matrix product and matrix equation systems. *SIAM J. Appl. Math.*, 17(3):603–606, 1969. (p 109)

[NFV93] N. Navab, O.D. Faugeras, and T. Viéville. The critical sets of lines for camera displacement estimation: A mixed euclidean-projective and constructive approach. In *Proceedings of the 4th International Conference on Computer Vision, Berlin, Germany*, pages 713–723. IEEE Computer Society Press, May 1993. (pp 31, 33, 34, 38)

[NNM96] A. K. Nayar, S. A. Nene, and H. Murase. Subspace methods for robot vision. *IEEE Trans. on Robotics and Automation*, 12(5):750–758, 1996. (p 17)

[NTG92] A. Nagchaudhuri, M. Thint, and D. P. Garg. Camera-robot transform for vision-guided tracking in a manufacturing work cell. *Journal of Intelligent and Robotic Systems*, 5:121–144, 1992. (p 20)

[Pen96] X. Pennec. *L'Incertitude dans les problèmes de reconnaissance et de recalage. Application en imagerie médicale et biologie moléculaire.* Thèse de doctorat, École Polytechnique., October 1996. (p 174)

[PL98] E. Panteley and A. Loria. On global uniform asymptotic stability of nonlinear time-varying systems in cascade. *Systems and Control Letters*, 33(2):131, 1998. (pp 60, 155)

[Plü65] J. Plücker. On a new geometry of space. *Philosophical Transactions of the Royal Society of London*, 155:725–791, 1865. (pp 31, 32, 160)

[PM94] Franck C. Park and Bryan J. Martin. Robot sensor calibration: Solving AX=XB on the euclidean group. *IEEE Trans. on Robotics and Automation*, 10(5):717–721, October 1994. (p 23)

[PMvdM98] M. Pagel, E. Maël, and Ch. von der Malsburg. Self calibration of the fixation movement of a stereo camera head. *Machine Learning*, 31:169–186, 1998. (p 19)

[PPR98] H. Pottmann, M. Peternell, and B. Ravani. Approximation in line space – applications in robot kinematics and surface reconstruction. In J. Lenarčič and M. L. Husty, editors, *Advances in Robot Kinematics: Analysis and Control*, pages 403–412. Kluwer Academic Publishers, 1998. (pp 32, 161)

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992. (pp 23, 24, 115, 143, 168)

[QK97] L. Quan and T. Kanade. Affine structure from line correspondences with uncalibrated affine cameras. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 19(8):834–845, August 1997. (p 13)

[RB89] F. Rotella and P. Borne. Explicit solution of Sylvester and Lyapunov equations. *Mathematics and Computers in Simulation*, 31:271–281, 1989. (p 108)

[RDLD97] S. Rémy, M. Dhome, J.M. Lavest, and N. Daucher. Hand-eye calibration. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1057–1065, 1997. (p 25)

[RE87] Patrick Rives and Bernard Espiau. Estimation recursive de primitives 3D au moyen d'une camera mobile. *Traitement du Signal*, 4:259–272, 1987. (p 38)

[Rém98] Sandrine Rémy. *Étalonnage d'un système de vision embarqué*. Thèse de doctorat, Université Blaise Pascal – Clermont II, Juillet 1998. (p 25)

[RH99] A. Ruf and R. Horaud. Visual servoing of robot manipulators, part i : Projective kinematics. Technical Report RR–3670, INRIA Rhône-Alpes, April 1999. (p 20)

[RL87] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*, volume XIV. John Wiley & Sons, Ltd., New York, 1987. (p 176)

[Rob95] L Robert. Camera calibration without feature extraction. *Computer Vision, Graphics and Image Processing*, 63(2):314–325, March 1995. also INRIA Technical Report 2204. (p 10)

[Rou97] P.J. Rousseeuw. Introduction to positive-breakdown methods. *Handbook of Statistics*, 15:101–121, 1997. (p 176)

[SA89] Y. C. Shiu and S. Ahmad. Calibration of wrist mounted robotic sensors by solving homogeneous transform equations of the form AX=XB. *IEEE Transactions on Robotics and Automation*, 5(1):16–29, February 1989. (pp 21, 22)

[SA90] M. Spetsakis and J. Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:171–183, 1990. (p 13)

[SBC94] V. Sundareswaran, P. Bouthemy, and F. Chaumette. Visual servoing using dynamic image parameters. Rapport de Recherche 2336, INRIA, Août 1994. (p 17)

[SC96] M. Spratling and R. Cipolla. Uncalibrated Visual Servoing. In *British Machine Vision Conference*, pages 545–554, 1996. (p 19)

[SK52] J.G. Semple and G.T. Kneebone. *Algebraic Projective Geometry*. Oxford Science Publication, 1952. (pp 32, 160)

[SK93] H. I. Suh and T. W. Kim. Visual servoing of robot manipulators by fuzzy membership function based neural networks. In Hashimoto [Has93]. (p 17)

[SLBE91] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control : The Task Function Approach* . Clarendon Press, Oxford, 1991. (pp 21, 35, 51, 81, 89, 118)

[SP94] S. Soatto and P. Perona. Structure-Independant Visual Motion Control on the Essential Manifold . In *Preprints of the Fourth IFAC Symposium on Robot Control*, September 1994. (p 17)

[SSV98] H. Sutanto, R. Sharma, and V Varma. The role of exploratory movement in visual servoing without calibration. *Robotics and Autonomous Systems*, 23:153–169, 1998. (p 18)

[SVS97] J. Santos-Victor and G. Sandini. Visual behaviors for docking. *Computer Vision and Image Understanding*, 67(3):223–238, 1997. (p 17)

[TL89] R. Y. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Trans. on Robotics and Automation*, 5(3):345–358, 1989. (pp 22, 114, 119, 123, 125, 130, 133)

[Ton97] Martin Tonko. *Zur sichtsystemgestützten Demontage am Beispiel von Altfahrzeugen.* PhD thesis, Universität Karlsruhe, 1997. (p 16)

[TRS97] D. Tsakiris, P. Rives, and C. Samson. Applying visual servoing techniques to control nonholonomic mobile robots. In *IEEE/RSJ/INRIA Workshop On New Trends in Image-based Robot Servoing*, pages 21–33, Grenoble, September 1997. (p 156)

[vALV94] G. D. van Albada, J. M. Lagerberg, and A. Visser. Eye in hand robot calibration. *Industrial Robot*, 21(6):14–17, 1994. (p 20)

[Vau85] Jacques Vaucanson. *Le Mécanisme du flûteur automate/ Jacques Vaucanson préf. Catherine Cardinal.* Archives contemporaines, Amsterdam, 1985. Réimpr. de l'éd. de Paris 1738. - Ed. bilingue français-anglais. (p 3)

[VIG01] VIGOR. Visually Guided Robots Using Uncalibrated Cameras, ESPRIT–IV reactive LTR project number 26247, 1998–2001. `http://www.inrialpes.fr/VIGOR`. (pp 6, 95)

[VLF95] T. Viéville, Q.T. Luong, and O.D. Faugeras. Motion of points and lines in the uncalibrated case. *International Journal of Computer Vision*, 17(1), 1995. (p 13)

[VY10] O. Veblen and J.W. Young. *Projective geometry.* The Athenaeum Press, 1910. (pp 162, 163)

[WAH98] G-Q Wei, K Arbter, and G Hirzinger. Active self-calibration of robotic eyes and hand-eye relationships with model identification. *IEEE Trans. on Robotics and Automation*, 14(1):158–165, 1998. (pp 25, 127)

[Wan92] C.-C. Wang. Extrinsic calibration of a robot sensor mounted on a robot. *IEEE Transactions on Robotics and Automation*, 8(2):161–175, April 1992. (p 116)

[WS97] Q. M. J. Wu and K. Stanley. Modular neural-visual servoing using a neural-fuzzy decision network. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3238–3243, 1997. (p 17)

[WSN87] L. E. Weiss, A. C. Sanderson, and C. P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, RA-3(5):404–417, October 1987. (p 16)

[WVT96] G. Wells, C. Venaille, and C. Torbas. Promising research vision-based robot positioning using neural networks. *Image and Vision Computing*, 14:715–732, 1996. (p 17)

[Xie97] Ming Xie. Robotic hand-eye coordination: new solutions with uncalibrated stereo cameras. *Machine Vision and Applications*, 10:136–143, 1997. (p 19)

[YA94] B.H. Yoshimi and P.K. Allen. Active, Uncalibrated Visual Servoing . In *Proc. IEEE International Conference on Robotics and Automation*, pages 156–161, 1994. (p 18)

[YA95] B.H. Yoshimi and P.K. Allen. Alignment using an uncalibrated camera system. *IEEE Trans. on Robotics and Automation*, 11(4):516–521, August 1995. (p 18)

[YCC95] Lu Y.-C. and J. C. K. Chou. Eight-space quaternion approach for robotic hand-eye calibration. In *International Conference on Systems, Man and Cybernetics*, volume 4, pages 3316–3321, 1995. (p 24)

[YW97] W.-Y. Yau and H. Wang. Robust hand-eye coordination. *Advanced Robotics*, 11(1):57–73, 1997. (p 17)

[Zha95] Z. Zhang. Estimating motion and stucture from correspondences of line segments between two perspective images. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 257–262. IEEE Computer Society Press, June 1995. (p 76)

[Zhu97] H. Zhuang. A note on "hand-eye calibration". *The International Journal of Robotics Research*, 16(5):725–727, October 1997. (p 24)

[Zhu98a] H. Zhuang. Hand/eye calibration for electronic assembly robots. *IEEE Trans. on Robotics and Automation*, 14(4):612–616, August 1998. (p 24)

[Zhu98b] H. Zhuang. A note on using only position equations for robotic hand/eye calibration. *IEEE Trans. on Systems, Man and Cybernetics*, 28(3):426–427, June 1998. (p 110)

[ZQ94] H. Zhuang and Z. Qu. A new identification jacobian for robotic hand/eye calibration. *IEEE Trans. on Systems, Man and Cybernetics*, 24(8):1284–1287, August 1994. (p 24)

[ZR91] H. Zhuang and Z. S. Roth. Comments on 'calibration of wrist mounted robotic sensors by solving homogeneous transform equations of the form AX=XB'. *IEEE Trans. on Robotics and Automation*, 7:877–878, December 1991. (p 24)

[ZR92] H. Zhuang and Z. S. Roth. Reply to "comments on 'comments on "calibration of wrist mounted robotic sensors by solving homogeneous transform equations of the form AX=XB"'". *IEEE Trans. on Robotics and Automation*, 8:493–494, August 1992. (p 24)

[ZRS94] H. Zhuang, Z. Roth, and R Sudhakar. Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation of the form AX=YB. *IEEE Trans. on Robotics and Automation*, 10(4):549–554, August 1994. (p 25)

[ZS93] H. Zhuang and Y. C. Shiu. A noise-tolerant algorithm for robotic hand-eye calibration with or without sensor orientation measurement. *IEEE Trans. on Systems, Man and Cybernetics*, 23(4):1168–1175, August 1993. (p 24)

[ZWR95] H. Zhuang, K. Wang, and Z. S. Roth. Simultaneous calibration of a robot and a hand-mounted camera. *IEEE Trans. on Robotics and Automation*, 11(5):649–660, October 1995. (pp 24, 25)

Asservissement visuel à partir de droites et auto-étalonnage pince-caméra

Résumé

L'utilisation de droites en asservissement visuel pose, contrairement au cas des points, un problème de représentation. Nous y avons répondu en nous basant sur les coordonnées de Plücker d'une droite, ce qui nous a permis d'introduire la notion d'alignement en coordonnées de Plücker binormées. Grâce à ces dernières, nous avons défini deux lois de commande voisines qui réalisent le nouvel alignement ; sont explicites et partiellement découplées entre rotation et translation ; mélangent informations 2D et 3D ; et enfin, ne nécessitent pas d'estimation de profondeur. Nous avons exhibé des résultats de convergence de ces lois et caractérisé leurs singularités. Nous avons ensuite appliqué ces lois au positionnement d'une caméra face à un trièdre orthogonal. Cette configuration ne permet pas d'observer la profondeur. Pour compenser ce manque, nous avons adjoint un pointeur laser non étalonné à la caméra.

En reformulant le problème d'étalonnage pince-caméra par un système purement linéaire, nous avons produit une analyse algébrique du système et une classification des mouvements d'étalonnage. Les procédures classiques sont contraignantes puisqu'elles nécessitent l'observation d'une mire et/ou l'interruption de la tâche effectuée par le robot. Afin de lever ces contraintes, nous avons adapté notre méthode linéaire pour proposer une méthode d'auto-étalonnage, qui se passe de mire, et une méthode d'étalonnage en ligne, qui n'interrompt pas la tâche.

Mots-clés — robotique, vision par ordinateur, asservissement visuel, droites, pointeur laser, étalonnage pince-caméra, auto-étalonnage, coordonnées de Plücker, équation de Sylvester, produit de Kronecker

Visual Servoing from Lines and Hand-Eye Self-Calibration

Abstract

Using lines in visual servoing yields, contrarily to the case of points, a representation choice. We solved it with the use of Plücker coordinates of a line. This allowed us to introduce the notion of binormalized Plücker coordinates alignment of lines. Using the latter coordinates, we derive two similar control laws that realize the new alignment, are explicit and partially decoupled in rotation and translation, mix 2D and 3D information, and finally do not require any depth estimation. We obtained convergence results for these laws and characterized their singularities. Then, we applied these laws to the positioning of a camera with respect to an orthognal trihedron. As this configuration is depth invariant, we added an uncalibrated laser beam to the camera.

With a new, entirely linear formulation of the hand-eye calibration paradigm, we produced an algebraic analysis and a classification of the calibration motions. The classical procedures require the use of a calibration block and/or to interrupt the robot task. To release these constraints, we adapted our linear formulation to form a self-calibration scheme, which gets rid of the calibration block, and an on-line calibration method, which does not interrupt the task.

Keywords — robotics, computer vision, visual servoing, lines, laser beam, hand-eye calibration, self-calibration, Plücker coordinates, Sylvester equation, Kronecker product

Thèse préparée dans les équipes Bip et Movi
des laboratoires inria Rhône-Alpes et gravir–imag
inria Rhône-Alpes, 655, av. de l'Europe, 38330 Montbonnot Saint Martin