De la vision artificielle à la réalité synthétique : système d'interaction avec un ordinateur utilisant l'analyse d'images vidéo

Daniel Dementhon

THESE

présentée par

Daniel DEMENTHON

pour obtenir le titre de DOCTEUR
de l'Université JOSEPH FOURIER – GRENOBLE I

(Arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)
(Spécialité Informatique)

De la Vision Artificielle à la Réalité Synthétique:

Système d'interaction avec un ordinateur utilisant l'analyse d'images vidéo

Thèse soutenue le 4 octobre 1993 devant la Commission d'examen.

Composition du Jury:

Président:	Claude PUECH
Rapporteurs:	Michel DHOME
	Olivier FAUGERAS
Examinateurs:	Camille BELLISSANT
	Radu HORAUD
	Jean-Sylvain LIENARD
	Annick MONTANVERT

Laboratoire TIMC/IMAG
URA CNRS D 1618

To my dearest, Margie, Eric and Louis.

Remerciements

D'abord je tiens à vivement remercier Olivier Faugeras et Michel Dhome pour avoir accepté d'être rapporteurs de cette thèse, ainsi que Claude Puech, Jean-Sylvain Liénard, Annick Montanvert et Radu Horaud, qui me font l'honneur de bien vouloir siéger dans le jury.

Je voudrais aussi exprimer ma profonde reconnaissance à tous ceux grâce à qui j'ai pu accomplir cette recherche. Je voudrais remercier tout particulièrement:

Camille Bellissant, qui m'a encadré, orienté et encouragé; cette collaboration électronique, grâce à l'Internet, *ftp* et PostScript, est sans doute un avant-goût de l'Université sans frontières du futur, où les étudiants rencontreront leur professeur en réalité virtuelle;

Larry Davis et Azriel Rosenfeld, qui m'ont fait confiance et m'ont donné leur appui alors que ce type de recherche n'était pas précisement dans les plans quinquénaux du Computer Vision Laboratory de l'Université du Maryland;

Yukio Fujii et Denis Oberkampf, pour leur collaboration et leur amitié durant leur année de visite aux Amériques; ils sont les co-auteurs de publications pour certaines parties de cette thèse;

Ben Shneiderman et Catherine Plaisant, du Computer Human Interface Laboratory de l'Université du Maryland, pour leurs encouragements et pour leurs conseils d'experts pour une utilisation confortable d'une souris 3D;

Thor Bestul, Yiannis Aloimonos, Dean Wierman, Vernon Weisenberg, Yaser Yacoob, Ansel Teng, Sunil Arya, et Sandy German, pour les discussions philosophiques, les objets en *C++*, les fibres optiques, *ftp*, les arbres binaires, les secrets de LaTeX, et surtout leur amitié.

Sommaire

Chapitre 1

Introduction

Nous décrivons un système qui permet d'interagir avec une scène virtuelle tridimensionnelle créée par un ordinateur. L'opérateur tient dans la main un pointeur (une "souris 3D") comportant une distribution de sources de lumière. Une caméra capture des images vidéo des sources de lumière, qui sont utilisées pour le calcul de la position et de l'orientation du pointeur à fréquence élevée. Nous décrivons les circuits qui détectent les images des sources de lumière dans chaque champ du signal vidéo, c'est-à-dire à 60 Hz. Nous présentons un nouvel algorithme, POSIT, développé spécialement pour cette application, qui utilise cette information pour calculer la pose du pointeur de manière très économique. Nous décrivons les opérations qui font le passage de la pose du pointeur à la représentation d'un curseur 3D dans la scène virtuelle, et qui minimisent les conséquences d'événements tels que la superposition ou la disparition d'images des sources de lumière. Nous analysons les métaphores qui permettent à l'opérateur d'interagir avec la scène de manière intuitive et efficace. Les méthodes que nous avons développées s'appliquent aussi au calcul de la pose de la tête de l'opérateur. Cette information permet à l'ordinateur de présenter des scènes qui varient en fonction du point de vue. Nous comparons notre système aux autres systèmes de poursuite utilisés dans le domaine de la réalité virtuelle.

1.1 RAMBO

En 1990, notre projet "RAMBO" (Robot Acting on Moving BOdies) s'était terminé sur un échec. Un bras de robot devait utiliser la vision pour suivre un objet polyhédrique se déplaçant de façon complexe (grâce à un deuxième bras de robot), et devait tenter de tirer sur certains points de l'objet avec un laser. Le côté "Guerre des Etoiles" du projet ne déplaisait pas à la DARPA, mais notre but était plutôt d'intéresser les gens de la NASA [10], qui venaient de manquer une tentative pour récupérer un satellite à partir de la navette spatiale; il devrait être possible de remplacer la technique "cow-boy" des astronautes par un système robotique équipé de caméras et capable de s'arrimer sur le satellite pour le ramener à bord. Nous avions d'abord une méthode initiale pour le calcul de la pose de l'objet avec une transformée de Hough pour trouver le vote de pose le plus élevé parmi des poses de triangles de points caractéristiques. Après quelques poses, on embrayait sur une méthode de poursuite visuelle basée sur un cycle de prédiction-correction dans l'image utilisant un filtre de Kalman [50]. A partir d'un calcul continu de poses, le robot aurait pu en principe contrôler sa propre trajectoire pour se positionner correctement par rapport à l'objet mobile. Mais il devint vite clair qu'une démonstration en temps réel était hors d'atteinte avec cette approche. D'abord, nos algorithmes faisaient confiance à la Connection Machine, mais il n'y avait pas de solution satisfaisante pour distribuer les pixels en temps réel parmi les processeurs de la machine. De plus, il était plus difficile d'obtenir un bon rendement de la Connection Machine que les GigaFLOPS des prospectus ne le laissaient prévoir. Notre calcul initial de pose prenait plusieurs secondes. Enfin, le système d'exploitation de notre bras de robot n'était pas adapté au contrôle en temps réel. Cette thèse est née en partie d'un désir de "venger RAMBO" en réussissant à faire tourner un système de calcul de pose en temps réel.

1.2 Fils de RAMBO

L'idée d'appliquer les concepts de RAMBO à une souris 3D nous est venue à la lecture d'un article de Byte, sur l'utilisation de gestes pour communiquer avec un ordinateur [39]; les

techniques nécessaires pour une souris 3D utilisant une caméra semblaient très similaires à celles développées pour RAMBO. Dans notre proposition pour un sujet de thèse, la Connection Machine était encore mentionnée pour résoudre le problème. Mais rapidement, les simplifications furent telles qu'on pouvait espérer tourner en temps réel sur un Macintosh.

1.3 Pose approchée rapide

On n'essaie pas ici d'interpréter une scène naturelle; on est libre de concevoir la souris 3D pour rendre la tâche la plus facile possible. En particulier, Steve Shaeffer à Carnegie Mellon University nous avait montré qu'avec une caméra noir et blanc on peut voir une main à travers un rideau de velours noir. Les capteurs CCD noir et blanc ont une grande partie de leur courbe de réponse en fréquence dans le domaine infrarouge et sont souvent équipées d'un filtre qui bloque les infrarouges. Il est facile de remplacer ce filtre par un filtre noir pour bloquer les longueurs d'onde de la lumière ambiante, et ne laisser passer que le rayonnement infrarouge. Les points caractéristiques de l'objet peuvent être constitués par des sources de lumière infrarouge facilement détectées dans l'image vidéo.

Pour simplifier la mise en correspondance entre ces sources de lumière et les taches claires qu'elles créent dans l'image vidéo, on peut soit utiliser le minimum de sources, soit utiliser des arrangements dont certaines caractéristiques sont invariantes ou varient peu en projection, tels que des alignements de sources dans des rapports de distance connus. Nous avons programmé les deux voies de cette alternative, mais préféré la première pour le prototype actuel.

Pour simplifier le calcul de pose une fois que la correspondance est connue, nous avions au départ appliqué les observations de Basri [55] pour une projection orthographique: le vecteur des coordonnées en x des points d'image d'un objet peut s'exprimer comme une combinaison linéaire entre vecteurs des coordonnées en x pour trois images prédéfinies, et cette combinaison linéaire est la même que pour le vecteur de la première ligne de la matrice de rotation de l'objet vis-à-vis des vecteurs des trois rotations utilisées pour les trois images; on trouvait donc la pose pour une nouvelle image en trouvant les coefficients de la combinaison linéaire entre les trois images prédéfinies. Mais l'équation finale montre que les trois images ne sont

pas nécessaires, et que le calcul revient à exprimer le vecteur des coordonnées en x comme produit d'une matrice caractérisant l'objet par la première ligne de la matrice de rotation. Cette égalité définit un système linéaire pour cette ligne de la matrice de rotation. Le calcul de la deuxième ligne utilise les coordonnées en y de la même manière, et la troisième ligne est le produit vectoriel des deux autres. Si on applique cette méthode à des images réelles et non des projections orthographiques, on trouve des lignes qui ne sont pas des vecteurs unitaires, mais leur norme est égale à l'échelle de la projection, liée à la distance focale et à la composante en z du vecteur de translation. Cette échelle permet de calculer la translation de l'objet.

Cette méthode fournit seulement une approximation de la pose, mais elle est extrêmement rapide, parce que dans les systèmes mentionnés ci dessus, une seule matrice est impliquée, qui caractérise la géométrie des points sur l'objet; son inverse ou pseudoinverse peuvent donc être calculés une fois pour toutes, et les opérations se résument à multiplier les vecteurs des coordonnées de l'image par cette matrice précalculée, puis à normaliser les vecteurs obtenus pour obtenir la rotation et la translation de l'objet. Cette méthode est présentée au chapitre 2.

1.4 Raffinement par itération

La pose obtenue par cette méthode approchée est utilisable pour un pointeur 3D, mais l'erreur de rotation entre le pointeur 3D et le curseur d'écran augmente à mesure que l'opérateur éloigne la souris 3D latéralement de l'axe optique de la caméra. On peut mettre en œuvre une méthode itérative très simple pour converger vers la pose réelle. L'idée est la suivante: dans la pose approchée, les points de l'objet ne tombent pas exactement sur les lignes de vue définies par les images de ces points (sinon la pose serait correcte!). On déplace les points pour les placer sur les lignes de vue, puis on projette ces points selon une projection orthographique à l'échelle. Cette image est généralement une meilleure projection orthographique à l'échelle de l'objet que l'image de départ, donc si maintenant on utilise cette image au lieu de l'image de départ, on obtient une meilleure pose, parce que le calcul de pose suppose précisemment un modèle de projection orthographique à l'échelle. De plus, on

montre que pour une séquence de calculs de poses appliqués à une séquence d'images, on peut même éviter les itérations si on utilise la pose trouvée pour la précédente image pour calculer les termes de correction (chapitre 2). Par rapport aux méthodes de calcul de pose utilisées couramment, cette méthode est très économique, et n'utilise pas de trigonométrie ni d'inversion de matrices. Même pour une machine rapide, ces considérations sont importantes parce que les calculs exigés par les animations tridimensionnelles interactives poussent en général les capacités de la machine à leur limite.

1.5 Points coplanaires

Lorsque les points caractéristiques de l'objet sont tous coplanaires, la matrice qui caractérise la distribution des points est singulière, et la méthode pour trouver la pose est un peu différente. L'approximation par projection orthographique à l'échelle fournit deux solutions symétriques, et pour chaque solution, on peut utiliser la même méthode itérative que pour des points non coplanaires. On obtient alors généralement deux solutions acceptables si la projection de l'objet sur l'axe optique est petite par rapport à la distance de la caméra, et une seule solution acceptable dans le cas contraire. Cette méthode est décrite au chapitre 3. Une configuration coplanaire de quatre sources ou plus a certains avantages; par exemple, il n'y a pas de risques de superpositions des images de sources à moins que le plan des sources contienne le centre de projection. Mais lorsque le plan fait presque face à la caméra, le choix entre les deux poses est difficile; d'autre part les calculs sont plus compliqués. Dans notre mise en application pour une souris 3D, nous avons préféré les configurations non coplanaires de sources.

1.6 Correspondance et pose

Enfin, pour terminer la partie théorique de cette thèse, nous montrons que les équations développées pour les poses rapides peuvent être utilisées pour la détermination simultanée des correspondances et de la pose, en présence de points d'image parasites et d'occlusions de points d'objet. Cette méthode a pour but de trouver la pose pour laquelle le plus grand nom-

bre de points de l'objet se projettent dans les carrés d'incertitude autour des points d'images. On cherche deux vecteurs caractérisant la pose en explorant simultanément deux arbres binaires pour subdiviser deux espaces à quatre dimensions, utilisant toutes les contraintes disponibles entre les deux espaces pour élaguer au maximum ces arbres. Nous espérions appliquer cette méthode à un pointeur comprenant une dizaine de sources réparties de manière aléatoire sur une sphère opaque, mais le temps de calcul (plusieurs secondes) nous en a dissuadé. Malgré tout, cette méthode mérite à notre avis sa place dans ce rapport car elle fournit une approche géometrique sans compromis et sans heuristique à un problème classique (chapitre 4). Nous espérons que d'autres chercheurs amélioreront cette technique pour la rendre assez rapide pour des applications en temps réel.

1.7 Réalité virtuelle et traqueurs

Ce projet nous a conduit à prêter attention aux recherches dans le domaine de la réalité virtuelle. La publicité donnée à la réalité virtuelle par les journaux et les films s'est considérablement amplifiée au cours des trois dernières années, parallèlement à l'effort de recherche et de commercialisation. Une des raisons est que le prix des ordinateurs capables de produire des animations graphiques interactives tridimensionnelles est tombé à un niveau plus accessible aux laboratoires. Ces animations interactives sont destinées à remplacer sur nos consoles l'univers plat des fenêtres, menus et icônes. L'enjeu est une productivité accrue, grâce à une meilleure capacité à interpréter des données abstraites lorsqu'elle sont codées sous forme visuelle en trois dimensions (chapitre 5).

La visualisation d'un écran avec des lunettes 3D fournit encore à l'heure actuelle une image très supérieure aux deux écrans d'un masque de visualisation, et cette approche plus confortable paraît satisfaisante pour de nombreuses applications (chapitre 5). Dans tous les cas, un système de poursuite ou "traqueur" est nécessaire pour transmettre à l'ordinateur les mouvements d'une souris 3D. Un autre traqueur transmettant le mouvement de la tête de l'opérateur permet à l'ordinateur de présenter des scènes qui dépendent du point de vue. Ce mécanisme est important même lorsqu'un écran fixe est utilisé, car il augmente l'effet 3D de manière considérable. La construction d'un traqueur de tête est similaire à celle

d'une souris 3D, et nous avons l'intention de construire un traqueur de tête optique selon les principes décrits dans cette thèse. Les méthodes optiques ne sont pas les seules possibles, et des traqueurs magnétiques, acoustiques et gyroscopiques sont disponibles sur le marché. Il nous a paru utile de décrire ces techniques et leurs points forts, et de fournir des points de comparaison avec le système que nous avons construit (chapitre 6).

1.8 Matériel et logiciel

Une des difficultés à surmonter avec un traqueur optique concerne l'analyse rapide des images. Avec des sources lumineuses, cette analyse consiste simplement à trouver les centres des taches claires créées par ces sources dans les images. Au début de ce projet, nous avions expérimenté sur un Macintosh avec une carte NuBus qui numérise les images vidéo et les place dans une mémoire où les pixels individuels sont accessibles. Le problème est qu'il faut trop de temps pour balayer cette mémoire à la recherche des taches claires. Si la carte pouvait marquer les lignes vides au moment de la mise en mémoire des données, le balayage serait beaucoup plus rapide. Mais ces cartes sont difficiles à modifier. Par contre, une série d'articles dans Byte en 1987 décrivait en détails les circuits et le fonctionnement d'un numériseur vidéo utilisant des composants courants. Nous sommes partis de cette base, que nous comprenions bien grâce aux articles, pour construire un détecteur rapide de taches claires. La collaboration de Yukio Fujii, un ingénieur de Hitachi en visite pour un an au labo, a été déterminante pour le succès de cette partie du projet (chapitre 7).

Au cours de cette recherche, nous avons découvert d'autres groupes travaillant sur la mise au point de traqueurs optiques: le groupe de l'Université de North–Carolina a un programme de recherche très avancé dans différents secteurs de la réalité virtuelle, et a développé des traqueurs de tête optiques. Parallèlement, les industries aéronautiques telles que Hughes et Honeywell ont développé des traqueurs optiques pour les casques des pilotes, pour permettre l'affichage d'information graphique superposée dans le champ visuel et la visée sans l'usage des mains; cette activité se reflète davantage dans les brevets d'invention que dans les journaux de recherche (chapitre 6). Pour résoudre la mise en correspondance, ces chercheurs éclairent les sources de lumière à tour de rôle, de façon qu'une seule source

soit visible dans chaque image. L'inconvénient de cette méthode est qu'il faut en principe obtenir autant d'images qu'il y a de sources avant de pouvoir calculer une pose; avec une caméra normale, la fréquence de calcul de pose devient trop lente, et pour cette raison ces chercheurs doivent utiliser des caméras spéciales rapides. Nous avons préféré laisser les sources éclairées en permanence et simplifier le problème de mise en correspondance en choisissant une configuration simple des sources. Ces techniques de mise en correspondance sont décrites en détail au chapitre 8. L'utilisation d'une caméra peu coûteuse est un facteur qui augmente les chances de succès d'une commercialisation.

Chapitre 2

POSIT, un Algorithme Rapide de Calcul de Pose

Dans ce chapitre, nous présentons une méthode pour calculer la pose d'un objet lorsqu'on voit dans l'image au moins quatre points caractéristiques non coplanaires et qu'on connaît leur géométrie relative sur l'objet. La méthode s'appuie sur deux algorithmes; un premier algorithme, *POS* (Pose from Orthography and Scaling), trouve la matrice de rotation et le vecteur de translation de l'objet par la résolution d'un système linéaire, en faisant une approximation de la perspective par une projection orthographique à l'échelle; un deuxième algorithme, *POSIT* (POS with ITerations), utilise dans sa boucle d'itération la pose approchée trouvée par POS pour calculer une meilleure projection orthographique à l'échelle de l'objet, puis applique POS à cette projection au lieu d'utiliser l'image de départ. POSIT converge vers une pose précise en quelques itérations. L'avantage principal de POSIT par rapport aux méthodes courantes est que le nombre d'opérations nécessaires est très faible, et qu'en particulier aucune inversion de matrice n'est nécessaire durant les calculs de pose.

2.1 Introduction

Le calcul de la position et de l'orientation d'un objet (la *pose* de l'objet) à partir d'images de points caractéristiques, lorsque la configuration géométrique de ces points est connue, a suscité de nombreux travaux de recherche, du fait que les applications sont nombreuses, pour la calibration, l'analyse d'images aériennes, la reconnaissance de formes et la poursuite d'objets en temps réel. Fischler et Bolles [19] ont introduit l'appellation "problème PnP" (*Perspective–n–Point* problem) pour ce type de problème lorsque n points caractéristiques sont utilisés.

Les chercheurs ont formulé des solutions analytiques pour des points caractéristiques dans des configurations coplanaires ou non coplanaires. Toutefois ces solutions ne sont pas nécessairement robustes. Par exemple, le problème P4P avec quatre points coplanaires a une seule solution analytique [1]. Toutefois, si les quatre points ne sont pas proches de la caméra, une pose qui est symétrique par rapport à un plan parallèle au plan de l'image se projette en une image qui est presque la même. Avec du bruit dans l'image, la solution analytique peut aussi bien tomber sur l'une ou l'autre pose, avec de bonnes chances de tomber sur la mauvaise pose.

Les méthodes de pose numériques peuvent utiliser un plus grand nombre de points que les méthodes de pose analytiques, et tendent à être plus robustes parce que les erreurs dues aux mesures et au bruit tendent à se compenser entre les différentes images de points, et parce que l'information implicite caractérisant la pose est redondante. Les méthodes proposées par Tsai [54] et par Yuan [63] sont typiques de telles méthodes numériques (ces papiers présentent aussi de bonnes revues des techniques de calibration photogrammétriques). Les méthodes proposées par Tsai sont spécialement utiles lorsque la distance focale et la distorsion de l'objectif sont inconnues. Une fois que ces quantités ont été calibrées, la méthode de Yuan peut suffire. Toutefois, ces méthodes ne sont pas bien adaptées au calcul de pose en temps réel, parce qu'elles font toutes deux appel pour la solution des équations finales à la méthode itérative de Newton-Raphson, qui exige des inversions de matrices à chaque pas d'itération.

La technique populaire introduite par Lowe [35, 36] a aussi pour base la méthode de Newton-Raphson. Elle suppose qu'une pose approchée a été découverte, soit par une autre

technique, soit par la même technique appliquée à l'image précédente si l'objet est poursuivi dans une séquence d'images. La méthode raffine cette pose approchée jusqu'à ce que les éléments caractéristiques (points ou lignes) de l'objet se projettent en coïncidence avec les images réelles de ces traits détectées dans l'image. Les fonctions que la méthode de Newton-Raphson fait tendre vers zéro sont les mesures des erreurs entre les projections calculées des éléments caractéristiques et leurs images réelles. Ces mesures sont exprimées en fonction des déplacements linéaires et angulaires de l'objet. A chaque pas d'itération, la méthode de Newton-Raphson calcule les déplacements qui réduisent ces mesures. Cela nécessite à chaque pas de recalculer l'inverse d'une matrice 6×6 construite à partir des valeurs numériques des dérivées de ces fonctions pour les déplacements calculés au pas d'itération précédent (cette matrice est un Jacobien si seulement trois points caractéristiques sont considérés).

La méthode présentée dans ce chapitre s'appuie sur des techniques d'algèbre linéaire comme la méthode de Yuan, et est itérative comme la méthode de Lowe. Par contre, la méthode proposée ne nécessite pas d'inversions de matrice durant le calcul de pose. Au premier pas de l'itération, la méthode trouve une pose approchée en composant une *matrice d'objet* précalculée, qui dépend uniquement de la distribution des points caractéristiques sur l'objet, et deux vecteurs, qui dépendent uniquement des coordonnées des images de ces points. Les deux vecteurs résultants, une fois normalisés, constituent les deux premières lignes de la matrice de rotation, et les normes de ces vecteurs fournissent la translation. Les itérations utilisent exactement ce même calcul, mais avec des points d'image corrigés par une opération très simple. La matrice d'objet mentionnée ci-dessus nécessite une inversion de matrice, mais ce calcul n'est effectué qu'une seule fois pour un objet donné, et cette matrice est réutilisée pour tous les calculs de pose concernant cet objet. Par conséquent, cette méthode de calcul de pose est très économique, et bien adaptée au problème de calcul de pose en temps réel tel qu'il se présente dans le type d'application considéré ici: le calcul de pose peut ainsi prendre une faible portion du temps de calcul de l'ordinateur, laissant le maximum de temps disponible pour la mise à jour continuelle d'images graphiques complexes.

2.2 Notations

Sur la figure 2.1, nous représentons le modèle classique de caméra à trou d'épingle, avec son centre de projection O, son plan d'image G à une distance f (la distance focale) de O, ses axes de repère Ox et Oy pointant dans la direction des lignes et colonnes du capteur de la caméra, et son axe Oz le long de l'axe optique. Les vecteurs unitaires pour ces trois axes sont appelés $\mathbf{i}$, $\mathbf{j}$ et $\mathbf{k}$ (les vecteurs sont écrits en caractères gras). La distance focale f, et le centre de l'image C à l'intersection de l'axe optique et du plan d'image, sont supposés connus.

Un objet, comprenant les points caractéristiques M_0, M_1, ..., M_i, ..., M_n, est situé dans le champ de la caméra. Le repère cartésien de l'objet est $(M_0\mathbf{u}, M_0\mathbf{v}, M_0\mathbf{w})$. Nous appelons M_0 le *point de référence* de l'objet. Les coordonnées (U_i, V_i, W_i) des points M_i dans le repère de l'objet sont connues. Les images de ces points M_i sont appelés m_i, et les coordonnées (x_i, y_i) de chacun des points d'image m_i sont connues. Par contre, les coordonnées (X_i, Y_i, Z_i) de ces mêmes points M_i dans le repère de la caméra sont inconnues, parce que la pose de l'objet dans ce repère est inconnue. Nous montrons dans la suite qu'on peut trouver la matrice de rotation et le vecteur de translation de l'objet sans chercher à calculer explicitement ces coordonnées (X_i, Y_i, Z_i).

2.3 Définition du problème

Notre but est de calculer la matrice de rotation et le vecteur de translation de l'objet dans le repère de la caméra. La matrice de rotation $\mathbf{R}$ de l'objet est la matrice dont les rangées sont les coordonnées des vecteurs unitaires $\mathbf{i}, \mathbf{j}, \mathbf{k}$ du repère de la caméra exprimées dans le repère de l'objet (M_0u, M_0v, M_0w). En effet, la matrice de rotation a pour rôle de transformer les coordonnées dans l'objet de vecteurs tels que $\mathbf{M_0M}_i$ en coordonnées définies dans le repère de la caméra; le produit scalaire $\mathbf{M_0M}_i \cdot \mathbf{i}$ entre la première rangée de la matrice et le vecteur $\mathbf{M_0M}_i$ fournit bien la projection de ce vecteur sur le vecteur unitaire $\mathbf{i}$ du repère de la caméra, c'est-à-dire la coordonnée $X_i - X_0$ de $\mathbf{M_0M}_i$ — pourvu que les coordonnées de $\mathbf{M_0M}_i$ et du vecteur-ligne $\mathbf{i}$ soient exprimées dans le même repère, ici le repère de l'objet. La matrice de

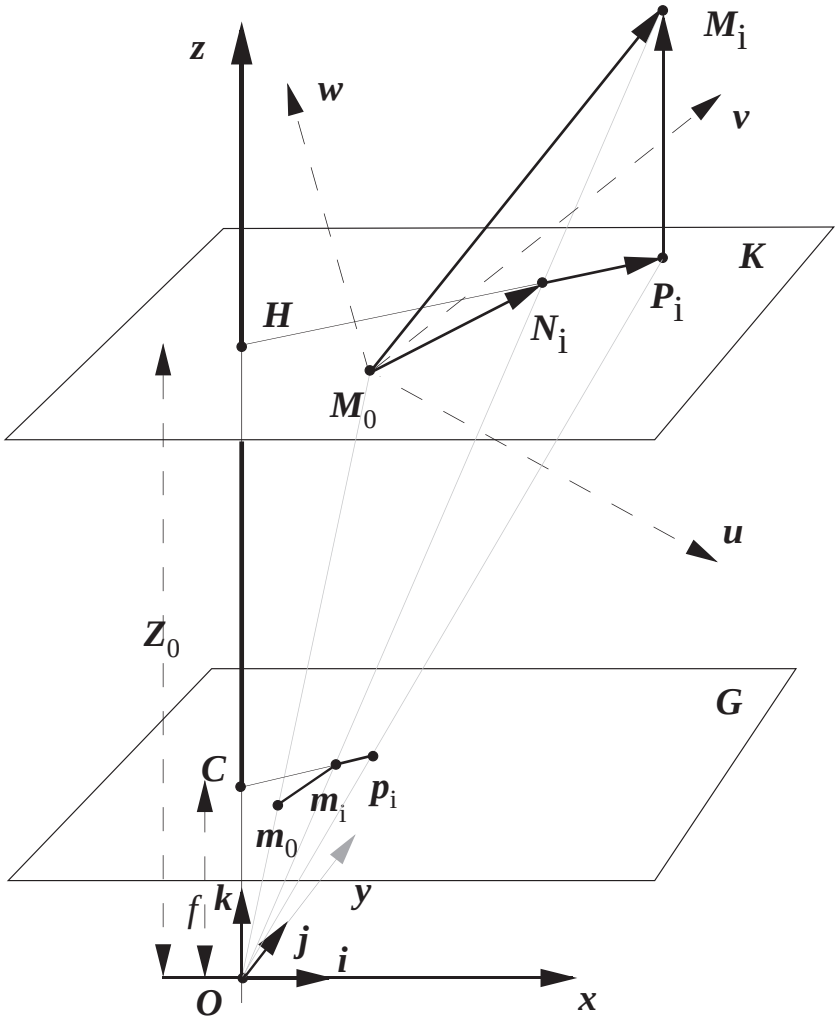

Figure 2.1: Projections perspectives (m_i) et projections orthographiques à l'échelle (p_i) pour les points M_i de l'objet

rotation peut donc s'écrire

$$\mathbf{R} = \begin{bmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{bmatrix}$$

Pour calculer la rotation, il suffit de calculer les vecteurs $\mathbf{i}$ et $\mathbf{j}$. Le vecteur $\mathbf{k}$ est déduit par le produit vectoriel $\mathbf{i} \times \mathbf{j}$. Le vecteur de translation, $\mathbf{T}$, est le vecteur $\mathbf{OM}_0$ entre le centre du repère de la caméra et le centre du repère de l'objet. Les coordonnées de ce vecteur sont donc X_0, Y_0, Z_0. Si on choisit comme centre du repère M_0 un des points caractéristiques visibles dans l'image, dont l'image est le point m_0, le vecteur de translation $\mathbf{T}$ est aligné avec

le vecteur $\mathbf{Om}_0$ et égal à $\frac{Z_0}{f}\mathbf{Om}_0$. Par conséquent pour calculer la translation de l'objet, il suffit de calculer sa coordonnée en z, Z_0. Pour résumer cette discussion, la pose de l'objet est complètement définie une fois que les inconnues $\mathbf{i}, \mathbf{j}$ et Z_0 ont été calculées. Ce chapitre est consacré spécifiquement au problème tel qu'il est défini ci-dessus, tandis que le chapitre 4 et l'annexe B présentent une formulation alternative qui ne nécessite pas la connaissance de l'image de M_0, ce qui est utile pour la mise en correspondance automatique des points d'objet et d'image. D'abord, nous rappelons quelques propriétés de la projection orthographique à l'échelle qui sont utilisées dans la suite.

2.4 Projection orthographique à l'échelle

2.4.1 Définition analytique

La projection orthographique à l'échelle (POE) est une approximation de la projection perspective "exacte" (PPE): pour un objet devant la caméra, on suppose que les profondeurs Z_i des points M_i de l'objet, dont les coordonnées dans le repère de la caméra sont (X_i, Y_i, Z_i), sont relativement peu différentes les unes des autres, et peuvent donc être remplacées par la profondeur Z_0 du point de référence M_0 de l'objet (voir figure 2.1). En POE, l'image d'un point M_i de l'objet est donc un point p_i du plan d'image G qui a pour coordonnées

$$x'_i = fX_i/Z_0, \quad y'_i = fY_i/Z_0,$$

alors qu'en PPE un point d'image m_i serait obtenu (au lieu de p_i), avec les coordonnées

$$x_i = fX_i/Z_i, \quad y_i = fY_i/Z_i$$

Le rapport $s = f/Z_0$ est le *facteur d'échelle* de la POE. Le point de référence M_0 a la même image m_0 et les mêmes coordonnées x_0 et y_0 en POE et en PPE.

Les coordonnées de la projection POE p_i de M_i peuvent aussi s'écrire

$$x'_i = fX_0/Z_0 + f(X_i - X_0)/Z_0 = x_0 + s(X_i - X_0)$$

$$y'_i = y_0 + s(Y_i - Y_0) \tag{2.1}$$

2.4.2 Construction géométrique

La construction géométrique pour obtenir l'image PPE, m_i, et l'image POE, p_i, de M_i est décrite sur la figure 2.1. L'image m_i est simplement l'intersection de la ligne de vue de M_i avec le plan d'image G. Par contre en POE, on trace le plan K passant par M_0 parallèlement au plan d'image G. Ce plan est à la distance Z_0 du centre de projection O. Le point M_i est projeté sur le plan K en P_i par une projection orthographique. Puis P_i est projeté sur le plan d'image G en p_i par une projection perspective. Le vecteur $\mathbf{m_0p}_i$ est parallèle au vecteur $\mathbf{M_0P}_i$ et est raccourci par rapport à $\mathbf{M_0P}_i$ par le facteur d'échelle s égal à f/Z_0. Les équations (2.1) expriment simplement la proportionnalité entre ces deux vecteurs.

2.5 Equations fondamentales

En projection perspective "exacte" des points M_i de l'objet, les équations de base qui relient les vecteurs lignes inconnus $\mathbf{i}$ et $\mathbf{j}$ de la matrice de rotation et la coordonnée Z_0 du vecteur de translation aux coordonnées connues des vecteurs $\mathbf{M_0M}_i$ dans le repère de l'objet, et aux coordonnées connues x_i and y_i des images m_0 et m_i de M_0 et M_i sont les équations suivantes

$$\mathbf{M_0M}_i \cdot \frac{f}{Z_0}\mathbf{i} = x_i(1+\varepsilon_i) - x_0, \tag{2.2}$$

$$\mathbf{M_0M}_i \cdot \frac{f}{Z_0}\mathbf{j} = y_i(1+\varepsilon_i) - y_0 \tag{2.3}$$

dans lesquelles ε_i est défini par

$$\varepsilon_i = \frac{1}{Z_0}\mathbf{M_0M}_i \cdot \mathbf{k}, \tag{2.4}$$

et $\mathbf{k}$ est défini par

$$\mathbf{k} = \mathbf{i} \times \mathbf{j} \tag{2.5}$$

Preuve: On considère sur la figure 2.1 les points M_0, M_i, de l'objet, le plan K parallèle au plan d'image et passant par M_0, l'intersection N_i de la ligne de vue M_i avec le plan K, et la projection P_i de M_i sur le plan K. Le vecteur $\mathbf{M_0M}_i$ est la somme de trois vecteurs

$$\mathbf{M_0M}_i = \mathbf{M_0N}_i + \mathbf{N}_i\mathbf{P}_i + \mathbf{P}_i\mathbf{M}_i \tag{2.6}$$

Le vecteur $\mathbf{M_0N}_i$ et son image $\mathbf{m_0m}_i$ sont proportionnels dans le rapport Z_0/f. Les deux vecteurs $\mathbf{N}_i\mathbf{P}_i$ et $\mathbf{Cm}_i$ sont aussi proportionnels dans les deux triangles semblables Cm_iO et $N_iP_iM_i$, dans un rapport égal au rapport des coordonnées en z des deux autres vecteurs correspondants de ces triangles, $\mathbf{P}_i\mathbf{M}_i$ et $\mathbf{OC}$. Ce rapport est $\mathbf{M_0M}_i \cdot \mathbf{k}/f$. La somme des vecteurs ci-dessus peut donc s'écrire

$$\mathbf{M_0M}_i = \frac{Z_0}{f}\mathbf{m_0m}_i + \frac{\mathbf{M_0M}_i \cdot \mathbf{k}}{f}\mathbf{Cm}_i + \mathbf{P}_i\mathbf{M}_i \tag{2.7}$$

On prend le produit scalaire de cette expression avec le vecteur unitaire $\mathbf{i}$ du repère de la caméra; le produit scalaire $\mathbf{P}_i\mathbf{M}_i \cdot \mathbf{i}$ est nul; le produit scalaire $\mathbf{m_0m}_i \cdot \mathbf{i}$ est la coordonnée en x, $x_i - x_0$, du vecteur $\mathbf{m_0m}_i$; le produit scalaire $\mathbf{Cm}_i \cdot \mathbf{i}$ est la coordonnée x_i de $\mathbf{Cm}_i$. Avec la définition $\varepsilon_i = \frac{1}{Z_0}\mathbf{M_0M}_i \cdot \mathbf{k}$, on obtient l'équation 2.2. De même, on obtient l'équation 2.3 en prenant le produit scalaire de l'expression 2.7 avec le vecteur unitaire $\mathbf{j}$ □

2.6 Equations fondamentales et POE

Nous montrons maintenant que les seconds membres des équations fondamentales, $x_i(1+\varepsilon_i) - x_0$ et $y_i(1+\varepsilon_i) - y_0$, sont en fait les coordonnées des points p_i, qui sont les POE des points caractéristiques M_i (figure 2.1):

Considérons les points M_0, M_i, la projection P_i de M_i sur le plan K, et son image p_i. Notons les coordonnées de p_i dans l'image x'_i et y'_i. Le vecteur $\mathbf{M_0M}_i$ est la somme de deux vecteurs $\mathbf{M_0P}_i$ et $\mathbf{P}_i\mathbf{M}_i$. Le premier vecteur, $\mathbf{M_0P}_i$, et son image $\mathbf{m_0p}_i$ sont proportionnels dans le rapport Z_0/f. Par conséquent

$$\mathbf{M_0M}_i = \frac{Z_0}{f}\mathbf{m_0p}_i + \mathbf{P}_i\mathbf{M}_i$$

On prend le produit scalaire de cette expression par le vecteur unitaire $\mathbf{i}$ du repère de la caméra; le produit scalaire $\mathbf{P}_i\mathbf{M}_i \cdot \mathbf{i}$ est nul, et le produit scalaire $\mathbf{m_0p}_i \cdot \mathbf{i}$ est la coordonnée en x, $x'_i - x_0$, du vecteur $\mathbf{m_0p}_i$. On obtient

$$\mathbf{M_0M}_i \cdot \frac{f}{Z_0}\mathbf{i} = x'_i - x_0 \tag{2.8}$$

et de même

$$\mathbf{M_0M}_i \cdot \frac{f}{Z_0}\mathbf{j} = y'_i - x_0, \tag{2.9}$$

Comparant ces équations avec les équations 2.2 et 2.3, on voit que les coordonnées de p_i sont en fait $x'_i = x_i(1+\varepsilon_i)$ et $y'_i = y_i(1+\varepsilon_i)$.

2.7 POS et POSIT

Les équations 2.2 et 2.3 peuvent encore s'écrire

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x_i(1+\varepsilon_i) - x_0, \tag{2.10}$$

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{J} = y_i(1+\varepsilon_i) - y_0, \tag{2.11}$$

avec

$$\mathbf{I} = \frac{f}{Z_0}\mathbf{i}, \quad \mathbf{J} = \frac{f}{Z_0}\mathbf{j}$$

L'idée de base de la méthode proposée est que si des valeurs sont données pour les termes ε_i, les équations 2.10 et 2.11 fournissent alors deux systèmes linéaires d'équations dans lesquels les seules inconnues sont respectivement les coordonnées de $\mathbf{I}$ et $\mathbf{J}$. Une fois que $\mathbf{I}$ et $\mathbf{J}$ ont été calculés, $\mathbf{i}$ et $\mathbf{j}$ en sont déduits par normalisation, et Z_0 est calculé à partir de la norme de $\mathbf{I}$ ou $\mathbf{J}$. Nous appelons l'algorithme de résolution de ces systèmes *POS* (Pose from Orthography and Scaling). En effet, chercher la pose de l'objet en utilisant les équations 2.2 et 2.3 avec des valeurs fixées pour ε_i revient à chercher la pose pour laquelle les points M_i ont pour POE les points p_i de coordonnées $x_i(1+\varepsilon_i)$ et $y_i(1+\varepsilon_i)$, comme on l'a vu dans la section précédente.

Les solutions de l'algorithme POS ne sont que des approximations si les valeurs données aux termes ε_i ne sont pas exactes. Mais une fois que les termes $\mathbf{i}$ et $\mathbf{j}$ ont été calculés, des valeurs généralement plus exactes peuvent être calculées pour les ε_i par les équations 2.5 et 2.4, et les deux systèmes linéaires peuvent être résolus à nouveau avec ces nouvelles valeurs de ε_i. Si aucune pose approximative de l'objet n'est disponible, on pose au départ $\varepsilon_i = 0$. Supposer ε_i nul implique $x'_i = x_i, y'_i = y_i$ et revient donc à supposer que p_i et m_i coïncident. La figure 2.2 décrit cette configuration. Si une pose a été calculée dans l'image précédente d'une séquence, on utilise cette pose pour calculer les valeurs initiales de ε_i. Nous appelons cet algorithme itératif *POSIT* (POS with Iterations). Cet algorithme fait généralement

converger les valeurs de $\mathbf{i}, \mathbf{j}$ et Z_0 vers les valeurs correspondant à une pose correcte en quelques itérations.

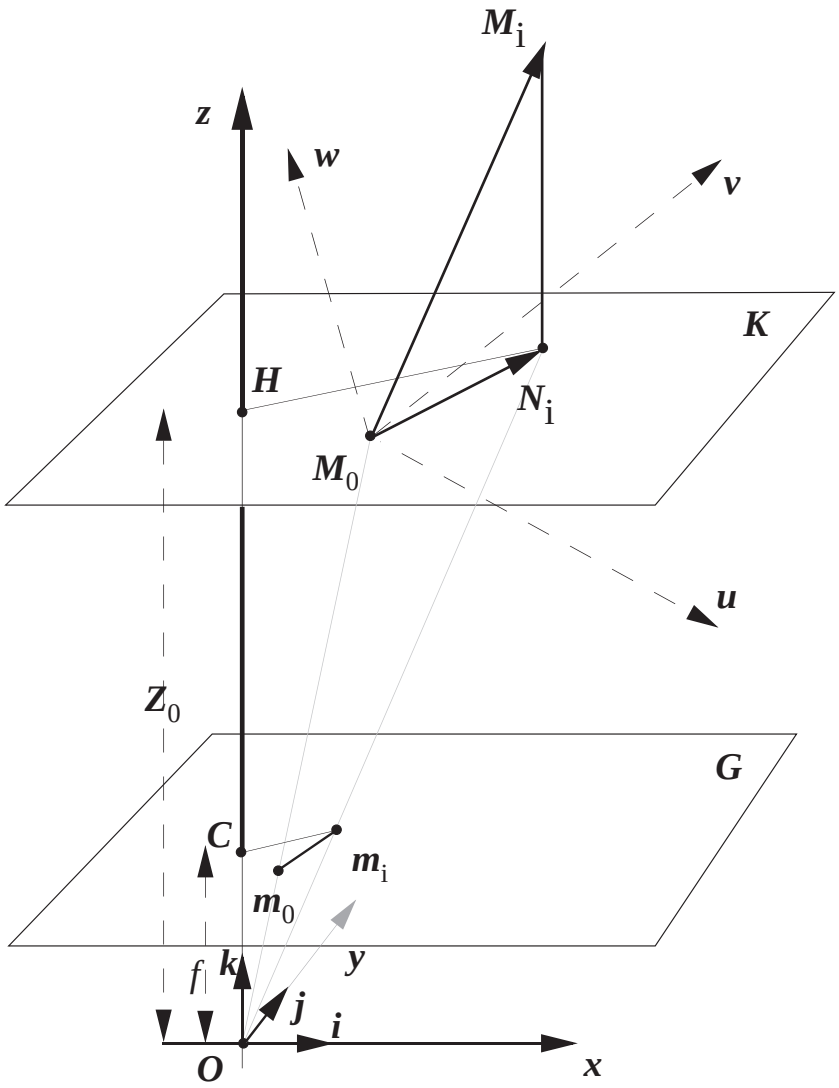

Figure 2.2: La boucle initiale de POSIT cherche la pose telle que les points m_i sont les projections orthographiques à l'échelle des points M_i de l'objet

2.8 Résolution des systèmes (algorithme POS)

Dans la boucle de l'algorithme itératif POSIT, on doit résoudre les équations 2.10 et 2.11. Nous les récrivons sous une forme un peu plus compacte

$$\mathbf{M_0M}_i \cdot \mathbf{I} = \xi_i,$$

$$\mathbf{M_0M}_i \cdot \mathbf{J} = \eta_i,$$

avec

$$\mathbf{I} = \frac{f}{Z_0}\mathbf{i}, \quad \mathbf{J} = \frac{f}{Z_0}\mathbf{j}, \quad \xi_i = x_i(1+\varepsilon_i) - x_0, \quad \eta_i = y_i(1+\varepsilon_i) - y_0,$$

et les termes ε_i ont des valeurs connues calculées aux pas précédents de l'algorithme. On exprime les produits scalaires de ces équations en termes des coordonnées des vecteurs exprimées *dans le repère de l'objet:*

$$[U_i\ V_i\ W_i][I_u\ I_v\ I_w]^T = \xi_i, \quad [U_i\ V_i\ W_i][J_u\ J_v\ J_w]^T = \eta_i,$$

où le T en exposant exprime le fait qu'on considère une matrice transposée, ici un vecteur colonne.

Ce sont des équations linéaires dans lesquelles les inconnues sont les coordonnées du vecteur $\mathbf{I}$ et du vecteur $\mathbf{J}$. Les autres termes sont connus: x_i, y_i, x_0, y_0 sont les coordonnées connues de m_i et m_0 (images de M_i et M_0) dans le repère de la caméra, U_i, V_i, W_i sont les coordonnées connues des points caractéristiques M_i dans le repère de l'objet, et les termes ε_i ont des valeurs calculées à chaque boucle d'itération.

Ecrivant les équations 2.10 et 2.11 pour les n points $M_1, M_2, M_i, \ldots, M_n$ et leurs images, nous obtenons des systèmes d'équations linéaires pour les coordonnées inconnues des vecteurs $\mathbf{I}$ et $\mathbf{J}$:

$$\mathbf{A\,I} = \mathbf{x}', \quad \mathbf{A\,J} = \mathbf{y}' \tag{2.12}$$

Ici $\mathbf{A}$ est la matrice des coordonnées des points M_i dans le repère de l'objet, $\mathbf{x}'$ est le vecteur dont la i-ème coordonnée est ξ_i et $\mathbf{y}'$ est le vecteur dont la i-ème coordonnée est η_i. En général, si on a au moins trois points visibles en plus de M_0, et si tous ces points sont *non*

coplanaires, la matrice $\mathbf{A}$ a le rang 3, et les solutions qui minimisent la somme des carrés des erreurs sont données par

$$\mathbf{I} = \mathbf{B}\,\mathbf{x}', \quad \mathbf{J} = \mathbf{B}\,\mathbf{y}' \tag{2.13}$$

où $\mathbf{B}$ est la matrice *pseudoinverse* de la matrice $\mathbf{A}$. Nous appelons $\mathbf{B}$ la *matrice d'objet*. Connaissant la distribution géométrique des points caractéristiques M_i, on peut *précalculer* cette matrice d'objet $\mathbf{B}$, soit par l'opération $[\mathbf{A}^T\mathbf{A}]^{-1}\mathbf{A}^T$, soit en décomposant $\mathbf{A}$ en valeurs singulières [47]. On peut trouver dans [47] une discussion montrant que les solutions calculées ainsi minimisent bien la somme des carrés des erreurs. Cette décomposition en valeurs singulières présente l'avantage de fournir un diagnostic clair au sujet du rang et de la condition de la matrice $\mathbf{A}$, ce qui peut être utile si on n'est pas sûr que les points caractéristiques sont bien non coplanaires (ce problème se poserait surtout en photogrammétrie). Dans le chapitre suivant, nous généralisons POSIT au cas des points coplanaires, et nous utilisons cette décomposition pour trouver la ou les poses de l'objet.

Une fois qu'on a obtenu les solutions au sens des moindres carrés pour $\mathbf{I}$ et $\mathbf{J}$, les vecteurs unitaires $\mathbf{i}$ et $\mathbf{j}$ sont obtenus simplement en normalisant les vecteurs $\mathbf{I}$ et $\mathbf{J}$. Comme on l'a indiqué dans la section 2.2, les trois éléments de la première ligne de la matrice de rotation sont les trois coordonnées du vecteur $\mathbf{i}$ obtenu ainsi. Les trois éléments de la seconde ligne sont les trois coordonnées du vecteur $\mathbf{j}$. Les éléments de la troisième ligne sont les coordonnées du vecteur unitaire $\mathbf{k}$ de l'axe des z du repère de la caméra et sont obtenus par le produit vectoriel des vecteurs $\mathbf{i}$ et $\mathbf{j}$.

A ce point on peut calculer le vecteur de translation de l'objet. C'est le vecteur $\mathbf{OM}_0$. Ce vecteur est aligné avec $\mathbf{Om}_0$ et égal à $Z_0\mathbf{Om}_0/f$, c'est-à-dire $\mathbf{Om}_0/s$. Le facteur d'échelle s est la norme du vecteur $\mathbf{I}$ ou du vecteur $\mathbf{J}$, comme l'indiquent les définitions de ces vecteurs en termes des vecteurs unitaires $\mathbf{i}$ et $\mathbf{j}$ données avec les équations 2.10 et 2.11.

2.9 Interprétation géométrique pour les solutions des systèmes

Géométriquement, l'équation 2.10 spécifie que si l'origine du vecteur $\mathbf{I}$ est prise au point M_0, l'extrémité de $\mathbf{I}$ se projette sur $\mathbf{M_0M}_i$ au point H_{xi} défini par la mesure algébrique

$$\overline{M_0H_{xi}} = \frac{\xi_i}{|\mathbf{M_0M}_i|}$$

Autrement dit, l'extrémité de $\mathbf{I}$ doit appartenir à un plan perpendiculaire à $\mathbf{M_0M}_i$ en H_{xi} (figure 3.1). Si on utilise quatre points caractéristiques, M_0, M_1, M_2, M_3, et si ces points sont choisis pour être non coplanaires, le vecteur $\mathbf{I}$ est complétement défini avec son origine en M_0 et son extrémité à l'intersection des trois plans respectivement perpendiculaires à $\mathbf{M_0M}_1$, $\mathbf{M_0M}_2$ et $\mathbf{M_0M}_3$ en H_{x1}, H_{x2} et H_{x3}. Analytiquement, on résoudrait un système de trois équations à trois inconnues, et la matrice $\mathbf{A}$ du système a le rang 3. On utiliserait dans ce cas pour matrice d'objet $\mathbf{B}$ l'inverse de la matrice $\mathbf{A}$ au lieu de la matrice pseudoinverse dans les équations 2.13.

Dès que plus de quatre points caractéristiques sont utilisés, les plans correspondants ne se coupent généralement pas en un seul point, mais on peut choisir comme extrémité de $\mathbf{I}$ le point pour lequel la somme des carrés des distances à ces plans est minimale. Analytiquement, le système d'équations est surdéterminé, la matrice est encore de rang 3, et la solution au sens des moindres carrés est obtenue en utilisant la matrice pseudoinverse $\mathbf{B}$ de la matrice du système $\mathbf{A}$ [47] dans les équations 2.13. Nous revenons sur cette interprétation géométrique dans la section 2.13 et dans les deux prochains chapitres.

2.10 Pseudo-code pour l'algorithme POSIT

Nous pouvons maintenant donner une description plus précise de l'algorithme itératif POSIT:

1. $\varepsilon_{i(0)} = 0, n = 1$
2. Début de la boucle.

   Résoudre pour les inconnues $\mathbf{i}, \mathbf{j}$, et Z_0 dans les systèmes décrits par les équations 2.2 et 2.3, avec les valeurs de ε_i trouvées au pas précédent (algorithme POS).

3. Calculer $\varepsilon_{i(n)} = \frac{1}{T_z}\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{k}$, avec $\mathbf{k} = \mathbf{i} \times \mathbf{j}$.

4. Si $|\varepsilon_{i(n)} - \varepsilon_{i(n-1)}| <$ Seuil, aller à l'étape 5.
   Sinon, $n = n + 1$; retourner à l'étape 2.

5. Pose exacte = dernière pose.

2.11 Interprétation géométrique de POSIT

L'algorithme itératif décrit analytiquement dans la section précédente peut aussi être décrit de manière géométrique [12]:

1. $p_{i(0)} = m_i$, $n = 1$.

2. Calculer une pose approchée de l'objet en supposant que les points M_i de l'objet se projettent aux points d'image $p_{i(n)}$ par une projection orthographique à l'échelle (algorithme POS).

3. Déplacer les points de l'objet à partir de leurs positions correspondant à la pose approchée calculée à l'étape précédente pour les placer sur les lignes de vue des images m_i tout en les laissant à la même profondeur (même Z) (cela correspond à une déformation de l'objet).
   Trouver les images $p_{i(n)}$ de ces points déplacés en utilisant une projection orthographique à l'échelle.

4. Si ces points POE sont à la même position que les points à l'itération précédente, aller à l'étape 5. Sinon retourner à l'étape 2.

5. Pose exacte = dernière pose.

Preuve: Notre but ici est de montrer que les étapes de calcul dans la description géométrique sont équivalentes aux étapes de calcul de la description analytique.

Etapes 1 et 2: Comme on l'a vu dans la section 2.6, chercher la pose en utilisant les équations 2.2 et 2.3 avec des valeurs calculées pour ε_i (description analytique) revient à

chercher la pose pour laquelle les points M_i ont pour POE les points p_i de coordonnées $x_i(1+\varepsilon_i)$ et $y_i(1+\varepsilon_i)$ (description géométrique). A l'étape 1, supposer ε_i nul (description analytique) implique $x'_i = x_i, y'_i = y_i$ et revient donc à supposer que p_i et m_i coïncident (figure 2.2).

Etape 3: Les points M_i de l'objet sont placés sur les lignes de vue des points m_i par un déplacement à Z constant puis projetés en p_i sur le plan d'image par une POE (description géométrique). Les coordonnées de p_i pour les points M_i déplacés sont $x_i(1+\varepsilon_i)$ et $y_i(1+\varepsilon_i)$, comme on vient de le voir dans l'examen de l'étape 2, avec $\varepsilon_i = \frac{1}{Z_0}\mathbf{M_0M}_i \cdot \mathbf{k}$ (équation 2.4). Ce produit scalaire et le terme Z_0 ne sont pas affectés par un déplacement de M_i perpendiculairement au vecteur $\mathbf{k}$, donc les termes ε_i peuvent être obtenus sans tenir compte de ce déplacement, en multipliant les $\mathbf{M_0M}_i$, définis une fois pour toutes dans le repère de l'objet, par le vecteur $\mathbf{k}$ dans le repère de l'objet, c'est-à-dire la troisième ligne de la matrice de rotation obtenue à l'étape 2 (description analytique).

Etape 4: Lorsque les termes ε_i) ne changent plus d'une itération à l'autre (description analytique), les expressions des coordonnées des points p_i ne changent plus, donc les positions des points p_i ne changent plus (description géométrique). L'un ou l'autre critère peut être utilisé. La description géométrique ne nécessite pas la définition d'un seuil artificiel si on sort de la boucle quand les positions discrètes des points p_i *en pixels* ne changent plus.

2.12 Interprétation intuitive de l'algorithme POSIT

Le processus qui conduit à une pose exacte avec l'algorithme POSIT peut être vu sous un troisième angle peut-être plus intuitif:

- Ce qui est connu est la distribution des points caractéristiques dans l'objet et les images de ces points par PPE.
- Si on pouvait construire les images POE, p_i, de ces points caractéristiques M_i (figure 2.1), on pourrait appliquer l'algorithme POS – qui résout un système linéaire et ne trouve une pose correcte que si les points d'image utilisés sont obtenus par POE – et on obtiendrait ainsi une pose exacte de l'objet.

- Mais pour connaître les images POE, on a besoin de la pose exacte de l'objet. Cependant on peut appliquer l'algorithme POS en utilisant les points d'image actuels m_i (figure 2.2). On obtient ainsi les profondeurs approchées pour les points caractéristiques, et on peut alors repositionner les points caractéristiques *à ces profondeurs, mais sur les lignes de vue*. On peut ensuite calculer les images POE de ces points. A l'étape suivante, on applique POS à ces images POE et on trouve généralement une pose plus précise, à partir de laquelle on peut calculer de meilleurs points d'image POE. En répétant ces itérations, on converge généralement vers des points d'image qui sont les points POE corrects des points caractéristiques, et donc vers une pose exacte.

2.13 Mesure d'erreur orthonormale

Pour une configuration non coplanaire de points caractéristiques, l'algorithme POS trouve les vecteurs $\mathbf{i}$ et $\mathbf{j}$ sans avoir recours aux conditions d'orthogonalité et d'égalité de normes entre ces deux vecteurs. On peut donc vérifier *a posteriori* si ces vecteurs satisfont ces contraintes. On peut en fait faire cette vérification avec les vecteurs $\mathbf{I}$ et $\mathbf{J}$, qui sont proportionnels à $\mathbf{i}$ et $\mathbf{j}$ par le même facteur d'échelle s. On définit pour cela une *mesure d'erreur orthonormale*, G, par exemple

$$G = |\mathbf{I} \cdot \mathbf{J}| + |\mathbf{I} \cdot \mathbf{I} - \mathbf{J} \cdot \mathbf{J}|$$

La mesure d'erreur orthonormale G est nulle quand les conditions d'égalité et d'orthogonalité sont vérifiées, et augmente avec la détérioration des résultats. Cette mesure peut être utilisée pour détecter les fausses correspondances entre les points caractéristiques de l'objet et les points d'image, et pour évaluer la qualité de détection des points d'image (voir chapitre 8).

Dans le cas idéal d'une détection sans erreurs de la position des images et d'une modélisation parfaite de la caméra par un modèle trou d'épingle, les équations 2.2 et 2.3 seraient vérifiées exactement si les termes ε_i correspondant à la pose réelle de l'objet sont utilisés. Dans l'interprétation géométrique de la section 2.9, les plans perpendiculaires aux vecteurs $\mathbf{M_0M}_i$ aux points H_{xi} définis aux abscisses $(x_i(1 + \varepsilon_i) - x_0)/|\mathbf{M_0M}_i|$ se couperaient exactement en un seul point correspondant à l'extrémité de $\mathbf{I}$ pour la pose de l'objet, et de

même, les plans perpendiculaires aux mêmes vecteurs aux points H_{yi} définis par les abscisses $(y_i(1+\varepsilon_i) - y_0)/|\mathbf{M}_0\mathbf{M}_i|$ se couperaient exactement à l'extrémité de **J** correspondant à la pose de l'objet. Pour ces deux vecteurs la mesure d'erreur serait nulle.

Durant le processus d'itération de l'algorithme POSIT, les termes ε_i sont d'abord initialisés à zéro, puis recalculés jusqu'à ce qu'ils correspondent à la pose réelle. Par conséquent, les points H_{xi} et H_{yi} sont initialement différents des points corrects, et se déplacent en direction des points corrects d'une itération à l'autre. Donc, initialement, les plans ne se coupent généralement pas en un point unique, et il y a peu de chance que les vecteurs **I** et **J** trouvés au cours des itérations intermédiaires à la distance RMS minimale entre ces plans puissent être égaux et perpendiculaires, et la mesure d'erreur orthonormale est élevée. A mesure que les points H_{xi} et H_{yi} tendent vers leurs positions correctes, la mesure d'erreur orthonormale tend vers zéro.

Ce scénario supposait une détection d'image et une modélisation de caméra parfaite. En pratique, les coordonnées x_i et y_i ne sont pas les coordonnées des projections perspectives idéales de M_i, et de ce fait, les plans obtenus au terme de la convergence ne se coupent généralement pas en des points uniques. En conséquence la mesure d'erreur orthonormale n'est généralement pas zéro (en fait, durant le processus d'itération, cette mesure passe souvent par une valeur plus faible que sa valeur finale, du fait que le critère de convergence utilisé n'est pas la mesure d'erreur orthonormale); la matrice de rotation finale n'est donc pas exactement orthogonale, mais comprend une composante de déformation qui voile l'objet de façon à l'ajuster aux lignes de vue d'une image déformée par les erreurs de détection. Dans la plupart des applications, une matrice de rotation orthonormale est plus utile. La matrice obtenue est facilement corrigée, par exemple en normalisant le vecteur **k** obtenu par le produit de **i** et **j**, puis en remplaçant **j** par le produit vectoriel de **k** et **i**.

2.14 Illustration du processus d'itération de POSIT

Nous illustrons le processus d'itération de POSIT par un exemple sur des données synthétiques. L'objet est un cube; les points caractéristiques sont les sommets du cube. La projection à gauche sur la figure 2.3 est l'image perspective de départ obtenue pour la pose réelle du

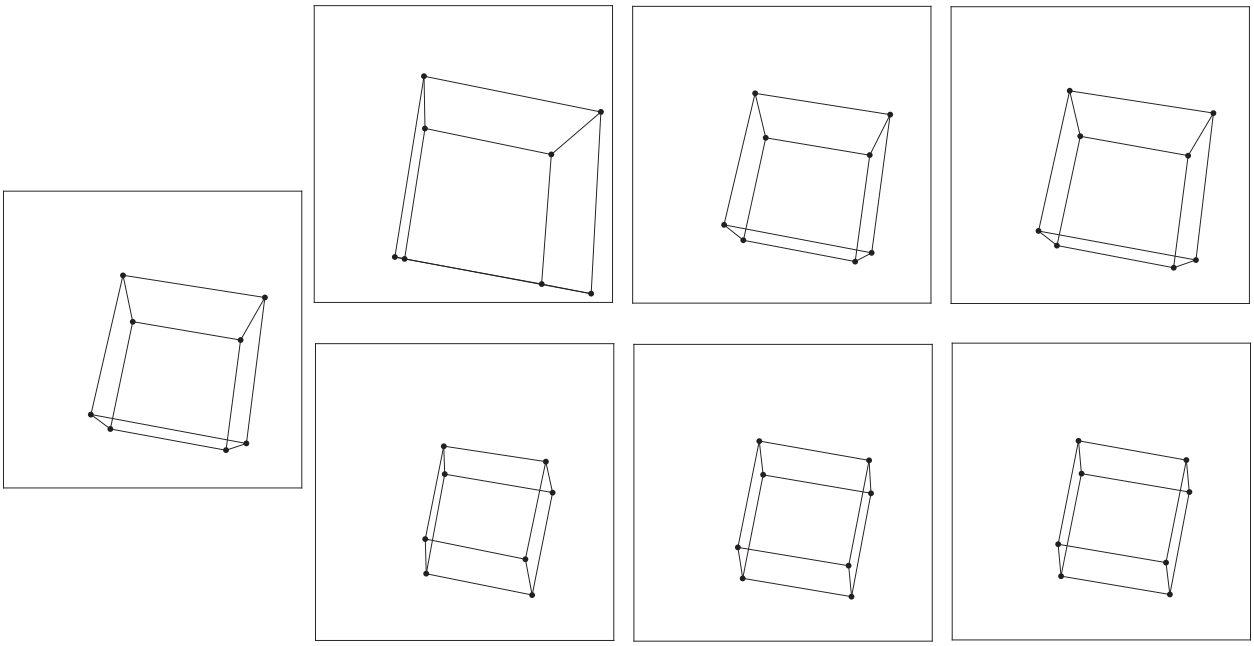

Figure 2.3: Images perspectives (rangée supérieure) et en projection orthographique à l'échelle (rangée inférieure) pour les poses du cube calculées dans les boucles successives de l'algorithme POSIT.

cube. Les projections des arêtes du cube qui sont tracées sur la figure ne sont pas utilisées par l'algorithme, mais sont utiles pour l'étude de l'évolution des projections orthographiques à l'échelle. Le rapport distance à la caméra/dimension est petit, par conséquent certaines arêtes présentent une forte convergence dans l'image perspective. On peut se faire une idée du succès de l'algorithme POSIT au cours de l'itération en calculant l'image perspective du cube pour les poses calculées et en comparant ces images avec l'image correspondant à la pose réelle (figure 2.3, rangée supérieure). On note que de gauche à droite ces projections deviennent de plus en plus semblables à l'image réelle. POSIT calcule les images de projection orthographique à l'échelle, représentées dans la rangée inférieure de la figure. On remarque que, de gauche à droite, les arêtes du cube deviennent de plus en plus parallèles dans ces images POE, une indication que l'algorithme parvient à calculer la projection orthographique correcte du cube, si on se rappelle que la projection orthographique préserve le parallélisme.

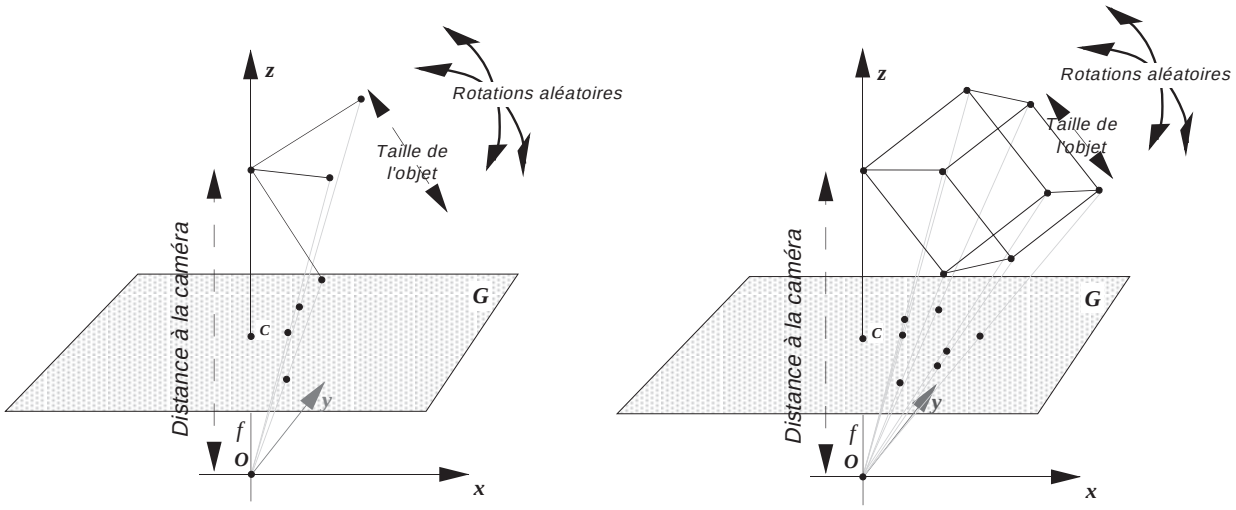

Figure 2.4: Définitions des objets et des paramètres pour les estimations des erreurs en rotation et en translation

2.15 Protocole de caractérisation de performance

Dans cette section, nous tentons de suivre les recommandations de Haralick pour l'évaluation de la performance des algorithmes de vision artificielle [28]. Nous calculons les erreurs en position et en orientation pour l'algorithme POS à la première boucle d'itération (une approximation de la projection perspective par une projection orthographique à l'échelle), et pour l'algorithme POSIT lorsqu'il a convergé. Des images synthétiques sont créées, et les poses calculées par POS et POSIT sont comparées avec les poses réelles qui ont servi à calculer ces images. Nous faisons ces calculs pour deux objets, à dix distances de la caméra, moyennant pour chaque distance les erreurs obtenues pour 40 orientations aléatoires. Nous répétons les expériences pour trois niveaux de bruit d'images.

2.15.1 Objets

Nous considérons deux objets:

1. Une configuration de quatre points (tétraèdre) tels que les trois segments joignant le point de référence aux trois autres points soient égaux (10 cm) et perpendiculaires (figure 2.4, gauche).

2. Les huit sommets d'un cube de 10 cm. Un des sommets est le point de référence (figure 2.4, droite).

2.15.2 Positions des objets

Le point de référence des objets est positionné sur l'axe optique. La distance du point de référence à la caméra est exprimée par rapport à la dimension caractéristique des objets (cette dimension est 10 cm). Dix distances sont considérées, de quatre à quarante fois la dimension d'objet. Ces distances relatives sont représentées en abscisses dans tous les graphes d'erreurs.

Autour de chacune des positions des points de référence le long de l'axe optique, les objets sont orientés dans 40 rotations. Les matrices de rotation sont calculées à partir de trois angles d'Euler choisis par un générateur de nombres aléatoires dans l'intervalle $(0, 2\pi)$. Les erreurs pour ces 40 rotations sont moyennées.

2.15.3 Génération des images

Nous obtenons les images des points caractéristiques par projection perspective avec une distance focale de 760 pixels. Nous ne coupons pas l'image, afin d'observer l'effet de grands décentrages des points d'images. Quand le point de référence du cube est à 40 cm du plan d'image sur l'axe optique et que le cube est complètement d'un côté de l'axe optique, le point sur la diagonale du cube peut se trouver à 30 cm du plan d'image et avoir une image à 355 pixels du centre de l'image. Il faut un objectif grand-angle avec un champ de vue de plus de 50 degrés pour voir tout le cube dans cette position.

Nous spécifions trois niveaux de bruit aléatoire dans l'image. Au niveau de bruit 1, les nombres réels calculés pour les coordonnées des projections perspectives des points sont arrondis à des nombres entiers définissant des positions discrètes de pixels.

Au niveau de bruit 2, ces positions sont déplacées par des perturbations aléatoires de $\pm$ 1 pixel, créées par un générateur de nombres aléatoires à distribution uniforme. Au niveau de bruit 3, l'amplitude de ces perturbations est $\pm$ 2 pixels. Quand l'objet est à 400 cm de la caméra, l'image de l'objet mesure environ 20 pixels, et une perturbation de 2 pixels de chaque côté de l'image représente une perturbation de 20% dans la dimension de l'image.

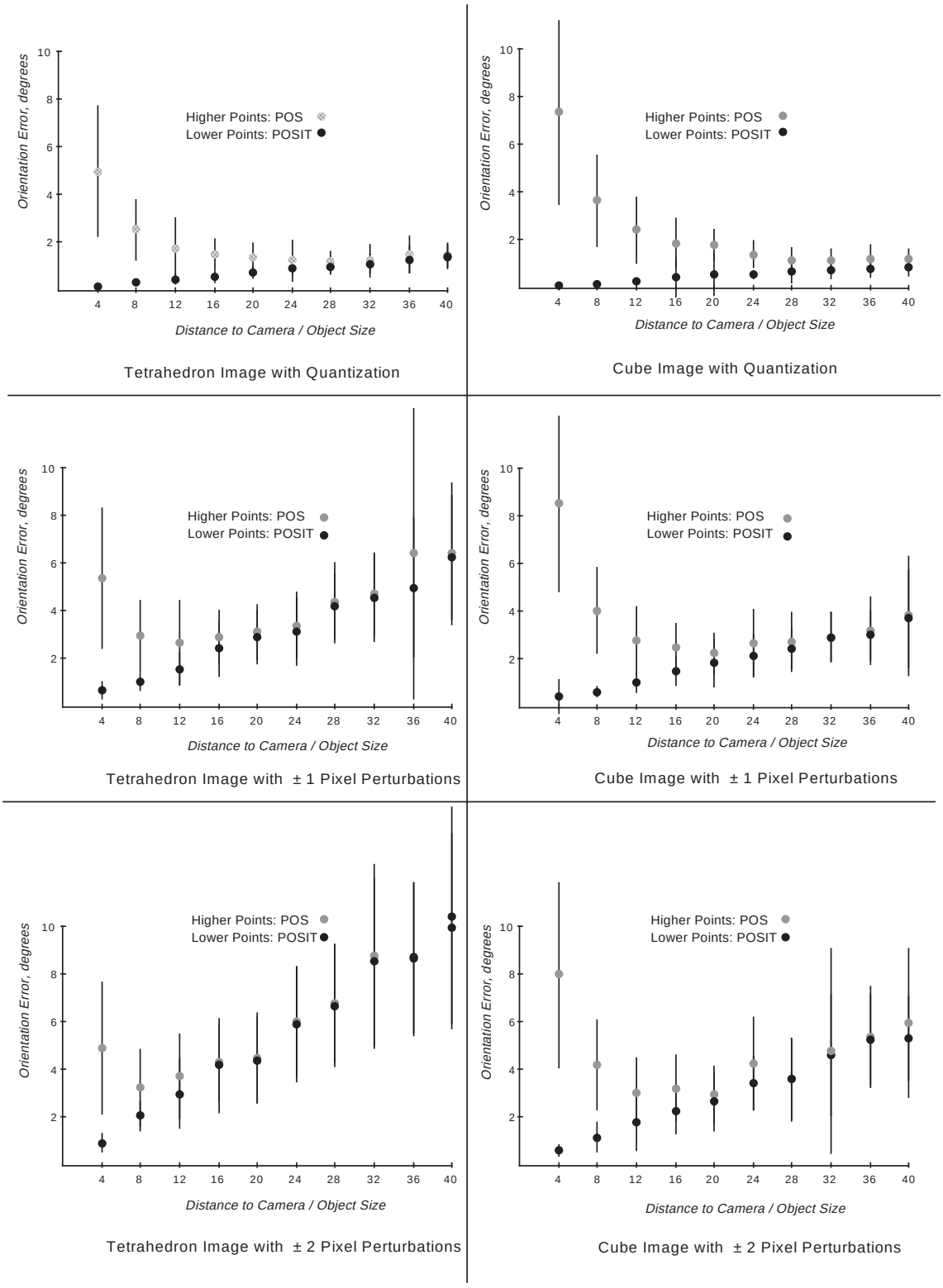

Figure 2.5: Erreurs angulaires d'orientation pour un tétraèdre (à gauche) et pour un cube (à droite) à 10 distances de la caméra, avec 3 niveaux de bruit d'image (discrétisation, ± 1 pixel, ± 2 pixels)

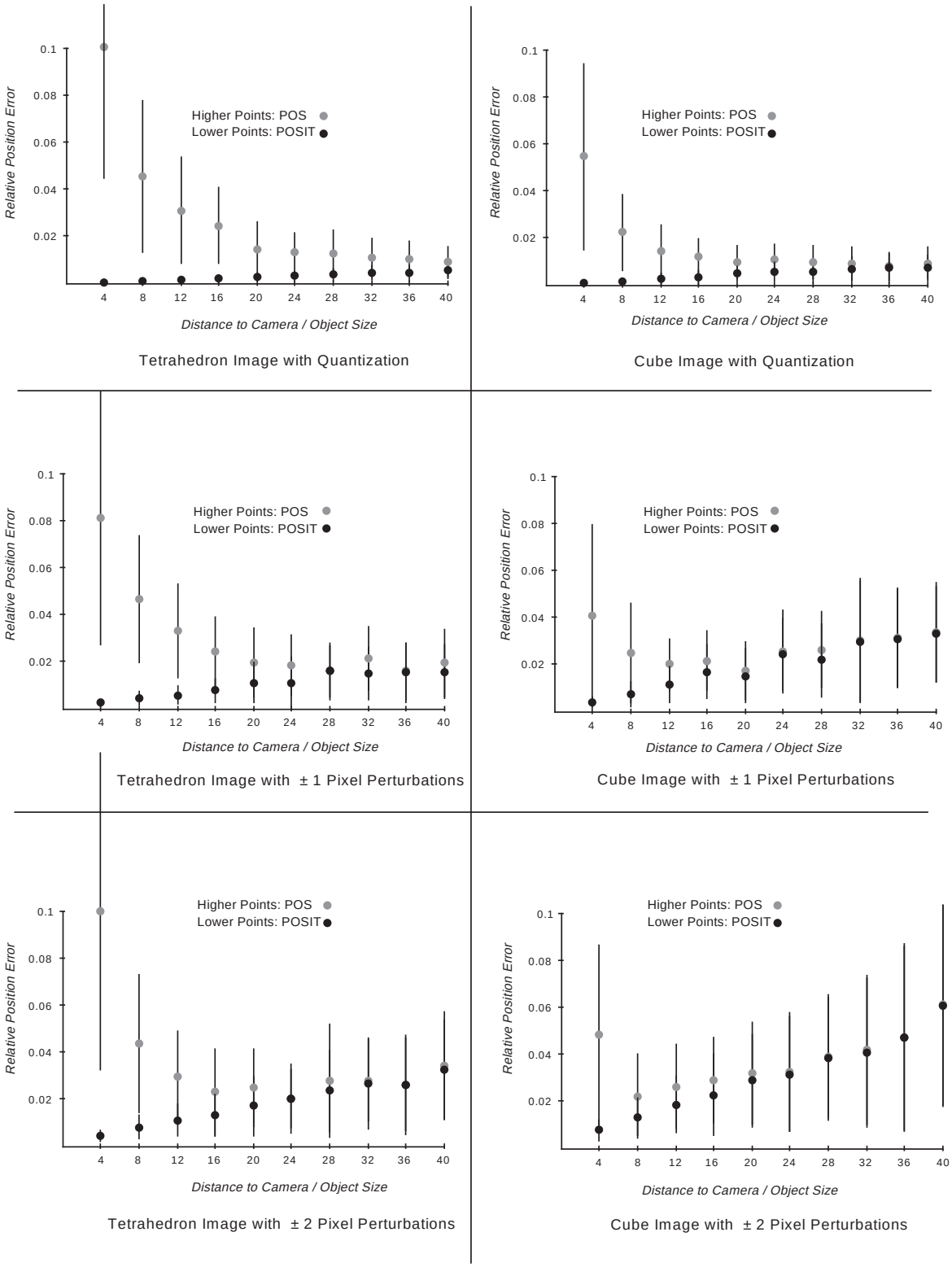

Figure 2.6: Erreurs relatives de position pour un tétraèdre (à gauche) et pour un cube (à droite) à 10 distances de la caméra, avec 3 niveaux de bruit d'image (discrétisation, ± 1 pixel, ± 2 pixels)

2.15.4 Calculs des erreurs d'orientation et de position

Pour chacune des images synthétiques, l'orientation et la position de l'objet sont calculées par l'algorithme POS (à la première boucle d'itération de POSIT, avec $\varepsilon_i = 0$), et par l'algorithme POSIT au terme de cinq boucles d'itération. Ces orientations et positions sont comparées à l'orientation et à la position de l'objet utilisées pour obtenir l'image. Pour la comparaison entre orientations, nous calculons l'axe de rotation nécessaire pour aligner le repère de l'objet dans sa position calculée avec le repère de l'objet dans sa position réelle. *L'erreur d'orientation* est définie comme l'angle de rotation en degrés autour de cet axe nécessaire pour produire cet alignement entre les deux repères. Les détails de calcul de l'axe et de l'angle de rotation sont fournis en annexe A. *L'erreur de position* est définie comme la norme du vecteur de translation nécessaire pour faire coïncider la position calculée du point de référence avec sa position réelle, divisée par la distance du point de référence réel à la caméra. L'erreur de position est donc une erreur relative, tandis que l'erreur d'orientation est exprimée en degrés.

2.15.5 Combinaison des résultats d'expériences multiples

Comme on l'a mentionné, pour chaque distance du point de référence de l'objet à la caméra, 40 orientations aléatoires de l'objet sont considérées. Nous calculons la moyenne et l'écart-type des erreurs d'orientation et de position pour toutes ces orientations, et nous représentons les points correspondant à ces moyennes avec leurs barres d'écart type en fonction des distances. Chaque diagramme comprend à la fois les points obtenus par POS, et à la cinquième itération. Les diagrammes pour les erreurs d'orientation sont représentés sur la figure 2.5, et les diagrammes pour les erreurs de position sont représentés sur la figure 2.6. Dans chacune de ces figures, les diagrammes de gauche sont pour le tétraèdre, et les diagrammes de droite pour le cube. Les diagrammes en haut des deux figures correspondent au niveau de bruit 1, ceux du milieu au niveau de bruit 2, et ceux du bas au niveau de bruit 3.

2.16 Analyse des diagrammes d'erreurs de pose

2.16.1 Comparaison entre POS et POSIT

Aux distances faibles et bruits faibles, POSIT produit des résultats qui semblent suffisamment précis pour les applications considérées, avec des erreurs d'orientation de moins de deux degrés et des erreurs de position de moins de 2%. Pour POSIT les erreurs augmentent linéairement avec la distance de l'objet à la caméra, à cause de la discrétisation des pixels de la caméra: on peut déplacer un point de l'espace deux fois plus lorsqu'il est deux fois plus loin de la caméra avant que le point d'image correspondant se déplace d'un pixel. Les effets du bruit d'image augmentent aussi avec la distance; à grande distance, les points d'image sont regroupés dans un espace de 20 pixels, et des perturbations de quelques pixels modifient la géométrie relative de ces points de façon significative.

POSIT produit une très nette amélioration des résultats par rapport à POS lorsque les objets sont très proches de la caméra, et pratiquement aucune amélioration lorsque les objets sont loin de la caméra. En effet, quand les objets sont proches de la caméra, les *distorsions de perspective* sont importantes, et la projection orthographique à l'échelle est dans ce cas une mauvaise approximation de la projection perspective. POS s'appuie sur cette approximation, et produit donc des résultats plutôt mauvais, avec des erreurs d'orientation autour de 10 degrés et des erreurs de positions autour de 10% pour le plus petit rapport distance/dimension. Par contre, quand les objets sont loin de la caméra, il y a peu de différence entre la projection perspective et la projection orthographique à l'échelle. Dans ce cas POS produit de bons résultats, et les itérations de POSIT produisent peu d'améliorations. POS donne les meilleurs résultats pour des distances d'objets entre 20 et 30 dimensions, à mi-chemin entre les fortes erreurs de distorsion de perspective aux courtes distances et les fortes erreurs dues à la discrétisation des pixels aux grandes distances.

2.16.2 Comparaison entre cube et tétraèdre

Les longues barres d'erreur pour l'algorithme POS semblent dues au fait que les dimensions de l'objet projeté dans l'image et les effets de perspective varient beaucoup en fonction de

l'orientation de l'objet. Le tétraèdre produit des changements de dimensions d'image moins importants que le cube, ce qui peut expliquer le fait qu'à faible distance les erreurs et écarts types produits par POS sont plus faibles pour le tétraèdre que pour le cube.

Avec un bruit élevé et de grandes distances, les erreurs sont deux fois moins élevées avec le cube qu'avec le tétraèdre, sans doute parce que POSIT fournit une solution moyenne dans laquelle les erreurs de chaque point ont plus de chances de se compenser lorsque deux fois plus de points sont utilisés.

2.17 Analyse de la convergence de POSIT

Avec les rapports distance/dimension utilisés dans les évaluations d'erreur de pose des sections précédentes, l'algorithme converge en quatre ou cinq itérations. On considère que la convergence est obtenue quand les positions en pixels des projections orthographiques à l'échelle des points caractéristiques aux poses calculées ne se déplacent plus d'une itération à la suivante. On peut appliquer POSIT à des images 1D dans un monde 2D, et dans ce cas il est possible de montrer que les facteurs déterminant la convergence sont les rapports des coordonnées d'image par la distance focale, c'est-à-dire les angles des lignes de vue avec l'axe optique. Quand ces rapports sont inférieurs à 1, l'algorithme converge. Les points caractéristiques sont alors à un angle de vue de moins de 45 degrés. Quand tous les points sont à plus de 45 degrés, l'algorithme diverge. Les points proches de l'axe optique ont tendance à faire converger l'algorithme et compensent l'influence des points loin de l'axe optique.

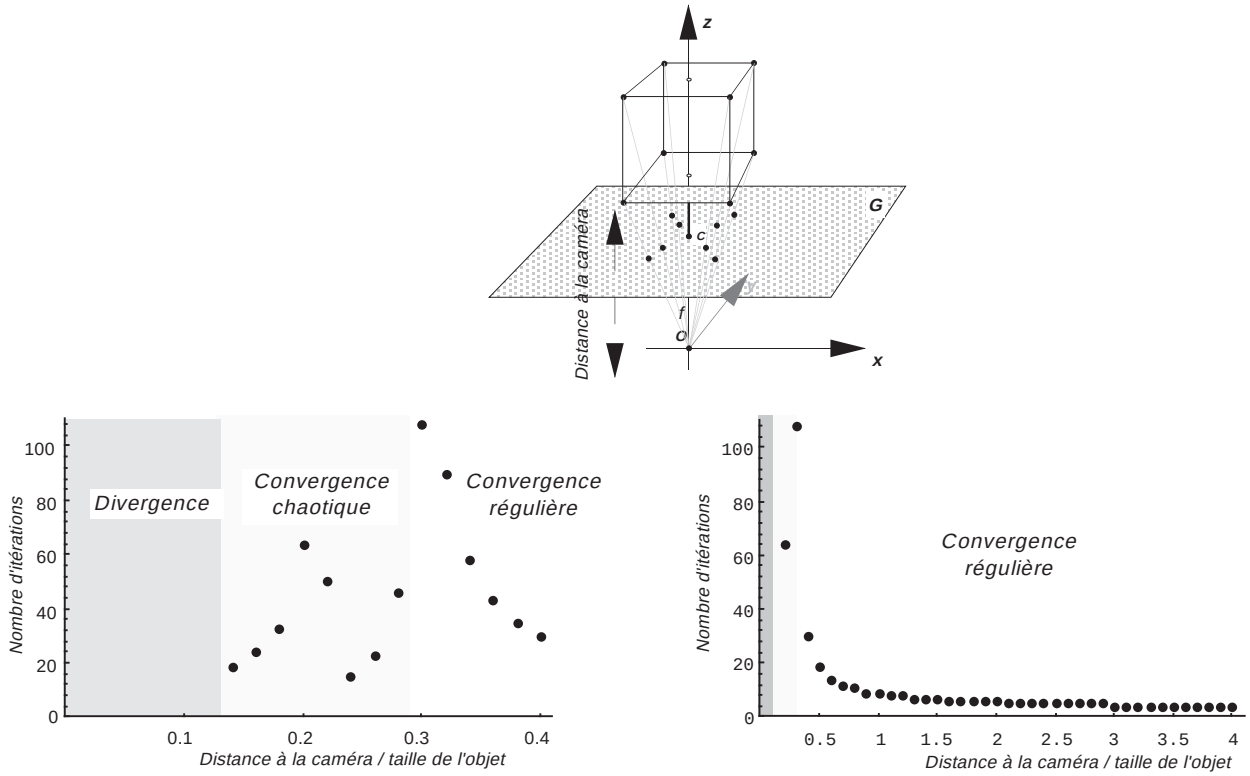

Figure 2.7: Nombre d'itérations pour POSIT en fonction de la distance de l'objet à la caméra. *En haut*: Définition de la distance de l'objet. *A gauche*: Analyse de la convergence aux courtes distances. La convergence apparaît si le cube de 10 cm est à plus de 1,2 cm de la caméra. *A droite*: Nombre d'itérations sur un domaine de distances plus étendu

Les simulations semblent démontrer des propriétés similaires avec des images 2D de points dans l'espace 3D. Dans ces simulations, un cube est déplacé le long de l'axe optique. On maintient deux faces parallèles au plan d'image, parce qu'aux plus proches distances de la caméra on ne peut changer l'orientation du cube sans entrer en collision avec le plan d'image. La distance utilisée dans le rapport distance/dimension des diagrammes est la distance entre le centre de projection et la face la plus proche. Un bruit de $\pm$ 2 pixels est ajouté aux projections perspectives des sommets du cube.

Pour un cube de 10 cm, quatre itérations sont nécessaires pour atteindre la convergence quand le cube est à plus de 30 cm du centre de projection de la caméra. Le nombre d'itérations augmente ensuite rapidement jusqu'à 100 quand le cube est approché jusqu'à une distance de 2.8 cm du centre de projection. En référence à nos observations précédentes sur les facteurs intervenant dans la convergence, les images des sommets les plus proches sont dans ce cas à plus de deux distances focales du centre de l'image, mais les images des sommets de la face la plus éloignée sont à une demi–distance focale du centre et contribuent probablement à préserver la convergence.

Jusqu'à ce point la convergence est monotone. A des distances encore plus proches, le mode devient non monotone, et les images orthographiques sont sujettes à des variations qui paraissent aléatoires d'une itération à l'autre, jusqu'à ce qu'une image parvienne à tomber près du résultat final, et la convergence est alors atteinte rapidement. Le nombre d'itérations est alors 20 à 60 dans ce mode, c'est-à-dire moins que dans le plus mauvais des cas pour le mode monotone, et ce nombre peut être très différent même si on déplace légèrement le cube. Pour cette raison nous appelons ce mode "chaotique" sur la figure 2.7.

Finalement, lorsque le cube est à moins de 1,2 cm du centre de projection, la différence entre images successives augmente rapidement, et l'algorithme diverge clairement. Il faut remarquer toutefois que le cube est alors pratiquement sur le plan d'image de la caméra et que dans la pratique il aurait été bloqué par l'objectif de la caméra avant d'atteindre cette position. De plus, pour que les sommets du cube soient visibles dans cette position, il faudrait que la caméra ait un champ de vue de plus de 150 degrés, c'est-à-dire une distance focale de 3 mm pour un capteur CCD de 20 mm. Les analyses préliminaires et ces expériences montrent que pour les modèles de caméra et les positions d'objet utiles dans la pratique,

l'algorithme semble converger sans problèmes en quelques itérations.

On sait toutefois que certaines configurations de points caractéristiques non coplanaires (nous examinons le cas des points coplanaires au chapitre suivant) peuvent avoir des images ambiguës pour certaines positions isolées de l'objet par rapport à la caméra pour une mise en correspondance donnée. L'image est ambiguë si des poses distinctes de l'objet dans l'espace peuvent produire la même image avec les mêmes mises en correspondance.[1] L'algorithme POSIT appliqué à une configuration non coplanaire de points produit une seule pose. Si l'objet se trouve à l'une des poses qui produisent une image ambiguë, il y a de bonnes chances que l'algorithme calcule une des autres poses produisant la même image, au lieu de la pose correcte. Nous notons pour notre défense que les autres algorithmes numériques, en particulier ceux qui convergent par la méthode de Newton-Raphson [35, 36] ne se comportent pas différemment dans ce cas. Il serait utile de pouvoir détecter au préalable la présence d'une image ambiguë, par exemple en construisant une matrice combinant les coordonnées d'image et d'objet et dont le déterminant s'annule dans ce cas. Nous réservons ce sujet à de futures recherches. Dans un mode de poursuite de l'objet pour lequel on traite une séquence d'images alors que l'objet se déplace, on peut en principe rejeter une mauvaise pose par comparaison avec les poses des images précédentes, du fait que les images ambiguës sont en principe rares et correspondent à des poses isolées.

2.18 En résumé

Nous avons présenté l'algorithme POSIT pour calculer la pose d'un objet à partir des images de points caractéristiques non coplanaires. Cet algorithme nécessite beaucoup moins de calculs que la plupart des méthodes courantes de pose. Nous avons présenté sous forme de pseudo-code les cinq étapes d'itération nécessaires pour le calcul, expliquant le rôle de chaque étape à la fois sous forme analytique et géométrique. L'algorithme calcule d'abord une approximation de la pose qui suppose que les images ont été obtenues par un modèle de projection othographique à l'échelle. Cette étape ne requiert pour obtenir la matrice de rota-

[1] Dans la suite, nous appelons aussi *image ambiguë* une image qui peut correspondre à plusieurs poses avec des mises en correspondance différentes du fait de symétries de l'objet.

tion que deux produits de vecteurs par une matrice précalculée, la normalisation des vecteurs résultants, et un produit vectoriel, et pour la matrice de translation la multiplication d'un vecteur par la norme utilisée dans la normalisation mentionnée. L'étape suivante consiste à remplacer les images réelles par des images orthographiques à l'échelle, en utilisant la pose approchée de l'étape précédente. Ces deux étapes sont répétées jusqu'à ce que les points d'images calculés et discrétisés en pixels ne se déplacent plus. Les tests de convergence montrent que l'algorithme converge en quelques itérations dans le domaine des configurations utiles de la caméra et de l'objet. Nous avons caractérisé la performance de l'algorithme par un grand nombre d'expériences sur des images synthétiques avec des niveaux croissants de bruit d'image; les erreurs sont typiquement de moins de 2% en translation et de moins de 2 degrés en rotation dans le domaine de distances qui nous intéresse. L'algorithme semble rester stable et se dégrader de manière prédictible lorsque le bruit augmente.

Chapitre 3

POSIT pour une Répartition Coplanaire de Points

Ce chapitre considère un objet avec une répartition *coplanaire* de points caractéristiques, et montre que l'algorithme POSIT présenté au chapitre précédent peut être appliqué avec quelques modifications. L'algorithme POSIT opère encore des itérations sur l'algorithme POS. Mais dans ce cas, l'algorithme POS fournit deux poses de l'objet. L'algorithme POSIT converge vers deux poses, et fournit pour ces deux poses une mesure de qualité. Lorsque l'objet est proche de la caméra, une des poses peut être clairement rejetée. Par contre, lorsque la distance de l'objet à la caméra est importante par rapport à la projection de l'objet sur l'axe optique, ou bien lorsque l'incertitude de position des images des points caractéristiques est élevée, les deux mesures de qualité sont similaires, et les deux poses sont alors des interprétations également plausibles de l'image de l'objet. Par comparaison, les méthodes utilisant une solution analytique pour quatre points coplanaires ne sont pas robustes pour les objets distants en présence de bruit d'image, parce qu'elles fournissent une seule pose parmi deux poses qui sont également possibles, et ont donc 50% de chances de fournir la mauvaise pose.

3.1 Introduction

Il est bien connu que le problème P3P (Perspective-3-Point problem) peut avoir jusqu'à quatre solutions [19, 30], bien que pour la plupart des positions de la caméra il n'y en a que deux [62]. Toutes ces solutions peuvent être exprimées comme solutions d'équations polynômiales de degré élevé [27]. Par contre, le problème P4P a une solution théorique unique [19, 63] si on exclut les cas où trois points sont alignés sur l'objet ou sur l'image. Les chercheurs ont formulé des solutions analytiques dans ce cas, pour des points caractéristiques non coplanaires [31] et pour des points coplanaires [1, 26].

Il est important de remarquer que la solution analytique unique proposée pour le problème P4P pour des points coplanaires ne peut être robuste quand la distance de l'objet à la caméra est importante par rapport à la projection de l'objet sur l'axe optique. En effet, dans cette configuration, on sait que la projection orthographique à l'échelle (POE) est une bonne approximation de la projection perspective "exacte" (PPE). Avec la POE, il y a toujours deux solutions de pose pour des points non coplanaires [11]. Ces deux poses sont symétriques par rapport à un plan parallèle au plan de l'image. En fonction du bruit dans l'image, la solution analytique peut basculer vers l'une ou l'autre pose, et a donc de bonnes chances de ne pas fournir la pose correcte de l'objet. Par conséquent, les méthodes analytiques qui fournissent une seule solution pour des points coplanaires ne sont pas dignes de confiance et devraient sans doute être évitées.

Ce chapitre présente un algorithme itératif qui est adapté à la fois aux cas des images prises près de l'objet et loin de l'objet. Cet algorithme est une version modifiée de l'algorithme POSIT du chapitre précédent. A partir de l'approximation POE, le processus d'itération trouve deux solutions acceptables. Lorsque l'objet est proche de la caméra, le processus d'itération trouve une seule solution possible. Comme la version du chapitre précédent, cet algorithme calcule les poses de l'objet sans inversion de matrices. Il est donc préférable pour des calculs en temps réel aux algorithmes issus de la méthode de Newton-Raphson. Nous fournissons une évaluation plus complète de cet algorithme dans [42].

3.2 Résolution des systèmes linéaires pour les points coplanaires

Les équations fondamentales 2.2 et 2.3 déterminées au chapitre précédent s'appliquent bien sûr dans le cas des points coplanaires. Ici aussi, ces équations sont résolues itérativement, en donnant au départ aux termes ε_i la valeur zéro, puis en opérant une boucle d'itération qui calcule les poses approchées comme solutions de systèmes linéaires et utilise ensuite ces poses pour calculer de meilleures estimations pour ε_i. La différence par rapport au chapitre précédent est la suivante: Du fait que les points sont coplanaires, la matrice $\mathbf{A}$, dont les lignes sont les coordonnées U_i, V_i, W_i des vecteurs $\mathbf{M_0M}_i$ dans le repère de l'objet, est de rang 2, puisque chaque ligne de la matrice est une combinaison linéaire de deux autres lignes. L'interprétation géométrique qui suit permet de trouver les solutions des systèmes dans ce cas.

3.3 Interprétation géométrique pour les solutions des systèmes

On se rappelle l'interprétation de l'équation 2.10: si l'origine du vecteur $\mathbf{I}$ est prise au point M_0, l'extrémité de $\mathbf{I}$ se projette sur $\mathbf{M_0M}_i$ au point H_{xi} défini par la mesure algébrique

$$\overline{M_0H_{xi}} = \frac{\xi_i}{|\mathbf{M_0M}_i|}$$

Autrement dit, l'extrémité de $\mathbf{I}$ doit appartenir à un plan perpendiculaire à $\mathbf{M_0M}_i$ en H_{xi}. Lorsque plusieurs vecteurs sont considérés, l'extrémité de $\mathbf{I}$ doit appartenir à l'intersection des plans correspondants. Supposons maintenant que les vecteurs $\mathbf{M_0M}_1$, $\mathbf{M_0M}_2$, , etc..., appartiennent à un plan D; les plans perpendiculaires à ces vecteurs en H_{x1}, H_{x2}, etc..., se coupent tous sur une seule ligne ou sur des lignes proches, parallèles entre elles et perpendiculaires au plan D. Le rang de la matrice $\mathbf{A}$ est 2 seulement; le système peut seulement définir le pied de l'intersection des plans perpendiculaires. La solution pseudo–inverse du système est un point Q du plan D pour lequel la distance aux plans perpendiculaires en H_{x1}, H_{x2},

etc..., est minimale, et on note le vecteur correspondant $\mathbf{I}_0$ (voir figure 3.1). Il est clair que cette solution n'est pas unique, puisque tous les vecteurs $\mathbf{I}$ dont l'extrémité se projettent sur le plan D en Q se projettent aussi sur $\mathbf{M}_0\mathbf{M}_1$ en H_{x1}, sur $\mathbf{M}_0\mathbf{M}_2$ en H_{x2}, etc.... Autrement dit, tout vecteur $\mathbf{I}$ dont l'origine est en $\mathbf{M}_0$ et l'extrémité sur la ligne perpendiculaire au plan D en Q est aussi une solution (Fig. 3.1).

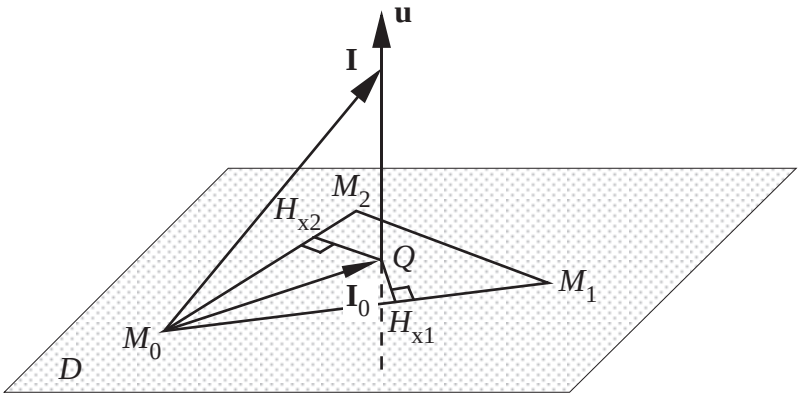

Figure 3.1: Tous les vecteurs $\mathbf{I}$ dont les extrémités se projettent sur le plan D en Q se projettent sur $\mathbf{M}_0\mathbf{M}_1$ en H_{x1} et sur $\mathbf{M}_0\mathbf{M}_2$ en H_{x2}

Une telle solution peut s'écrire

$$\mathbf{I} = \mathbf{I}_0 + \lambda\mathbf{u} \tag{3.1}$$

où $\mathbf{u}$ est le vecteur unitaire normal au plan D et λ est la coordonnée de l'extrémité de $\mathbf{I}$ le long de $\mathbf{u}$. De même

$$\mathbf{J} = \mathbf{J}_0 + \mu\mathbf{u} \tag{3.2}$$

3.4 Calcul des solutions I et J

Puisque le vecteur $\mathbf{u}$ défini dans la section précédente est normal au plan D de l'objet, on a $\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{u} = 0$. Le vecteur $\mathbf{u}$ est donc le vecteur unitaire de l'espace nul de la matrice $\mathbf{A}$. Il peut être obtenu comme le vecteur colonne correspondant à la plus petite valeur singulière dans la deuxième matrice orthonormée fournie par la décomposition en valeurs singulières $\mathbf{A}$ [47]. Ce calcul de $\mathbf{u}$ fournit une direction moyenne, et paraît donc utile dans le cas où les points ne sont pas exactement coplanaires. Ce calcul est effectué une seule fois pour une distribution de points coplanaires donnée, en même temps que le calcul de la matrice objet

$\mathbf{B}$. Dans les équations 3.1 et 3.2, les vecteurs $\mathbf{I}_0$ et $\mathbf{J}_0$ sont des solutions du système qu'on obtient en écrivant $\mathbf{I}_0 = \mathbf{B}\,\mathbf{x}'$, et $\mathbf{J}_0 = \mathbf{B}\,\mathbf{y}'$; $\mathbf{x}'$ et $\mathbf{y}'$ sont définis dans la section 2.8. Les paramètres λ et μ sont des inconnues; nous devons donc dans ce cas utiliser les contraintes supplémentaires spécifiant que $\mathbf{I}$ et $\mathbf{J}$ doivent être perpendiculaires et de même longueur de manière à calculer λ et μ. La première contrainte impose

$$\lambda\,\mu = -\mathbf{I}_0 \cdot \mathbf{J}_0,$$

et la deuxième contrainte impose

$$\lambda^2 - \mu^2 = \mathbf{J}_0^2 - \mathbf{I}_0^2$$

Huttenlocher [32] trouve ces mêmes deux équations, dans le cas particulier de trois points, par une démarche très différente, et les résout en élevant la première équation au carré et en éliminant une des inconnues entre les deux équations. Cette procédure semble introduire des solutions supplémentaires qui ne sont pas solutions des équations de départ et doivent être éliminées. Nous proposons ici une méthode qui ne nécessite pas l'élévation au carré de la première équation.

On remarque que le carré du nombre complexe $C = \lambda + i\mu$ est $C^2 = \lambda^2 - \mu^2 + 2i\lambda\mu$, c'est-à-dire

$$C^2 = \mathbf{J}_0^2 - \mathbf{I}_0^2 - 2i\mathbf{I}_0 \cdot \mathbf{J}_0$$

On peut donc trouver λ et μ comme les parties réelle et imaginaire des deux racines du nombre complexe C^2. On trouve ces racines en écrivant C^2 sous forme polaire:

$$C^2 = [R, \Theta], \text{ avec :}$$

$$R = ((\mathbf{J}_0^2 - \mathbf{I}_0^2)^2 + 4(\mathbf{I}_0 \cdot \mathbf{J}_0)^2)^{1/2},$$

$$\Theta = \text{Arctan}(\frac{-2\mathbf{I}_0 \cdot \mathbf{J}_0}{\mathbf{J}_0^2 - \mathbf{I}_0^2}), \text{ si } \mathbf{J}_0^2 - \mathbf{I}_0^2 > 0, \text{ et}$$

$$\Theta = \text{Arctan}(\frac{-2\mathbf{I}_0 \cdot \mathbf{J}_0}{\mathbf{J}_0^2 - \mathbf{I}_0^2}) + \pi, \text{ si } \mathbf{J}_0^2 - \mathbf{I}_0^2 < 0$$

$$(\text{si } \mathbf{J}_0^2 - \mathbf{I}_0^2 = 0, \text{ on a } \theta = -\text{Sign}(\mathbf{I}_0 \cdot \mathbf{J}_0)\frac{\pi}{2}, \text{ et } R = |2\mathbf{I}_0 \cdot \mathbf{J}_0|)$$

Les deux racines de C^2 sont $C_1 = [\rho, \theta]$, et $C_2 = [\rho, \theta + \pi]$, avec

$$\rho = \sqrt{R}, \text{ et } \theta = \frac{\Theta}{2}$$

λ et μ sont les parties réelles et imaginaires de ces nombres:

$$\lambda = \rho \cos\theta, \quad \mu = \rho \sin\theta, \text{ ou bien}$$

$$\lambda = -\rho \cos\theta, \quad \mu = -\rho \sin\theta$$

Ces paires de valeurs produisent deux solutions pour les paires ($\mathbf{I}$, $\mathbf{J}$)

$$\mathbf{I} = \mathbf{I}_0 + (\rho \cos\theta)\mathbf{u}, \quad \mathbf{J} = \mathbf{J}_0 + (\rho \sin\theta)\mathbf{u}, \text{ et}$$

$$\mathbf{I} = \mathbf{I}_0 - (\rho \cos\theta)\mathbf{u}, \quad \mathbf{J} = \mathbf{J}_0 - (\rho \sin\theta)\mathbf{u}$$

On note que, du fait que $\mathbf{u}$ est perpendiculaire au plan de l'objet, la paire de vecteurs $(\mathbf{I}, \mathbf{J})$ de la première solution est la symétrique par rapport à ce plan de la paire $(\mathbf{I}, \mathbf{J})$ de la seconde solution. Si on se rappelle que $\mathbf{I}$ et $\mathbf{J}$ sont parallèles aux vecteurs unitaires $\mathbf{i}$ et $\mathbf{j}$ du repère de la caméra, on voit que les deux solutions pour la pose approchée du plan de l'objet sont symétriques par rapport à un plan $(\mathbf{I}, \mathbf{J})$, c'est-à-dire symétrique par rapport à un plan parallèle au plan de l'image (figure 3.2).

3.5 De I et J aux poses approchées

A ce point, on utilise les deux solutions pour $\mathbf{I}$ et $\mathbf{J}$ pour trouver le vecteur de translation et les matrices de rotation par une démarche similaire à celle de la section 2.8. On obtient la coordonnée Z_0 du vecteur de translation en divisant la distance focale par la norme d'un des vecteurs $\mathbf{I}$ ou $\mathbf{J}$ (on a imposé la condition $|\mathbf{I}| = |\mathbf{J}|$). On trouve donc une solution $T_z = Z_0$ unique. Puis on obtient $T_x = \frac{T_z}{f} x_0$, $T_y = \frac{T_z}{f} y_0$. La solution pour le vecteur de translation est donc unique. Par contre, on obtient deux matrices de rotation correspondant aux deux paires de solutions $\mathbf{I}$ et $\mathbf{J}$; pour chaque paire, la normalisation de $\mathbf{I}$ et $\mathbf{J}$ produit les deux premières

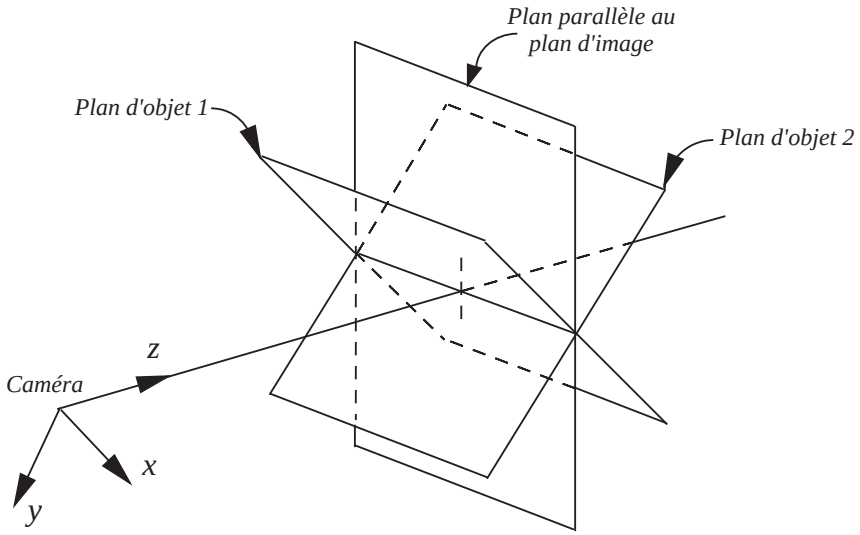

Figure 3.2: Deux poses d'un objet plan produisant la même image par une projection orthographique a l'échelle

rangées de la matrice, et la dernière rangée est le produit vectoriel des deux premières). Ces deux solutions correspondent aux deux positions symétriques du plan de l'objet par rapport au plan parallèle au plan d'image et passant par la position du point de référence M_0 de l'objet, qui est unique puisque la translation est unique (figure 3.2).

Toutefois, ces deux poses ne sont pas toujours physiquement possibles. On doit vérifier que pour chacune, tous les points de l'objet se trouvent bien devant la caméra, c'est-à-dire que toutes les coordonnées en z, Z_i, de ses points sont positives. Si ce n'est pas le cas, la pose correspondante est rejetée.

3.6 Itérations vers les poses exactes

On vient de voir que pour les objets plans, l'algorithme POS produit deux poses à chaque itération de l'algorithme POSIT. Il semblerait donc que nous devions explorer un arbre de poses nous conduisant à 2^n poses après n itérations (voir figures 3.3 et 3.4 pour une illustration de ces arbres). Toutefois en pratique on se rend compte qu'on doit suivre seulement une ou deux branches, et on trouve soit une seule solution, soit deux solutions. En effet une des deux situations suivantes peut se produire:

- Dans la première situation (figure 3.3), on calcule deux poses au premier pas d'itération, mais on découvre que l'une des poses doit être rejetée parce que certains points sont placés derrière la caméra dans cette pose. Cette situation se produit ensuite pour toutes les itérations suivantes. Par conséquent, dans ce cas on ne peut suivre qu'une seule voie et on ne trouve qu'une pose physiquement possible.

- Dans la deuxième situation (figure 3.4), les deux poses du premier pas d'itération sont possibles, mais pour chaque pas *on ne garde que la meilleure des deux poses*. Cette stratégie se justifie par le fait que l'exploration d'une branche de moins bonne qualité n'est pas utile: les expériences montrent que si on explore une telle branche, soit on retrouve l'une des deux poses, mais avec un taux de convergence beaucoup plus faible (figure 3.4), soit le processus d'itération diverge.

Pour choisir la meilleure des deux poses pour chaque itération après la première itération, on utilise comme mesure de qualité la mesure de distance E, distance moyenne entre points d'image actuels et points projetés pour les poses calculées.

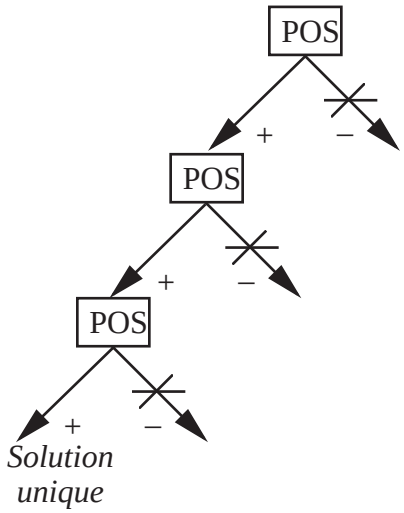

Figure 3.3: Cas 1: POS produit une seule pose possible à chaque niveau d'itération (+: pose possible, −: pose impossible)

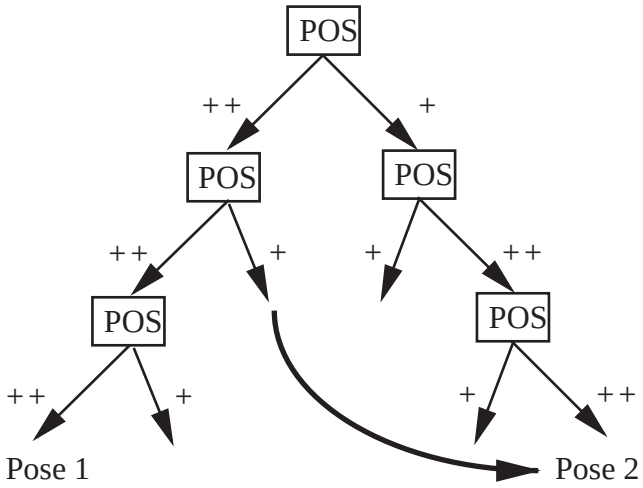

Figure 3.4: Cas 2: POS produit deux solutions possibles à chaque niveau d'itération (++: pose de meilleure qualité, +: pose de moindre qualité)

L'organigramme pour l'algorithme POSIT pour points coplanaires est représenté sur la figure 3.5. Il montre les deux branches produites à la première itération. Une des branches

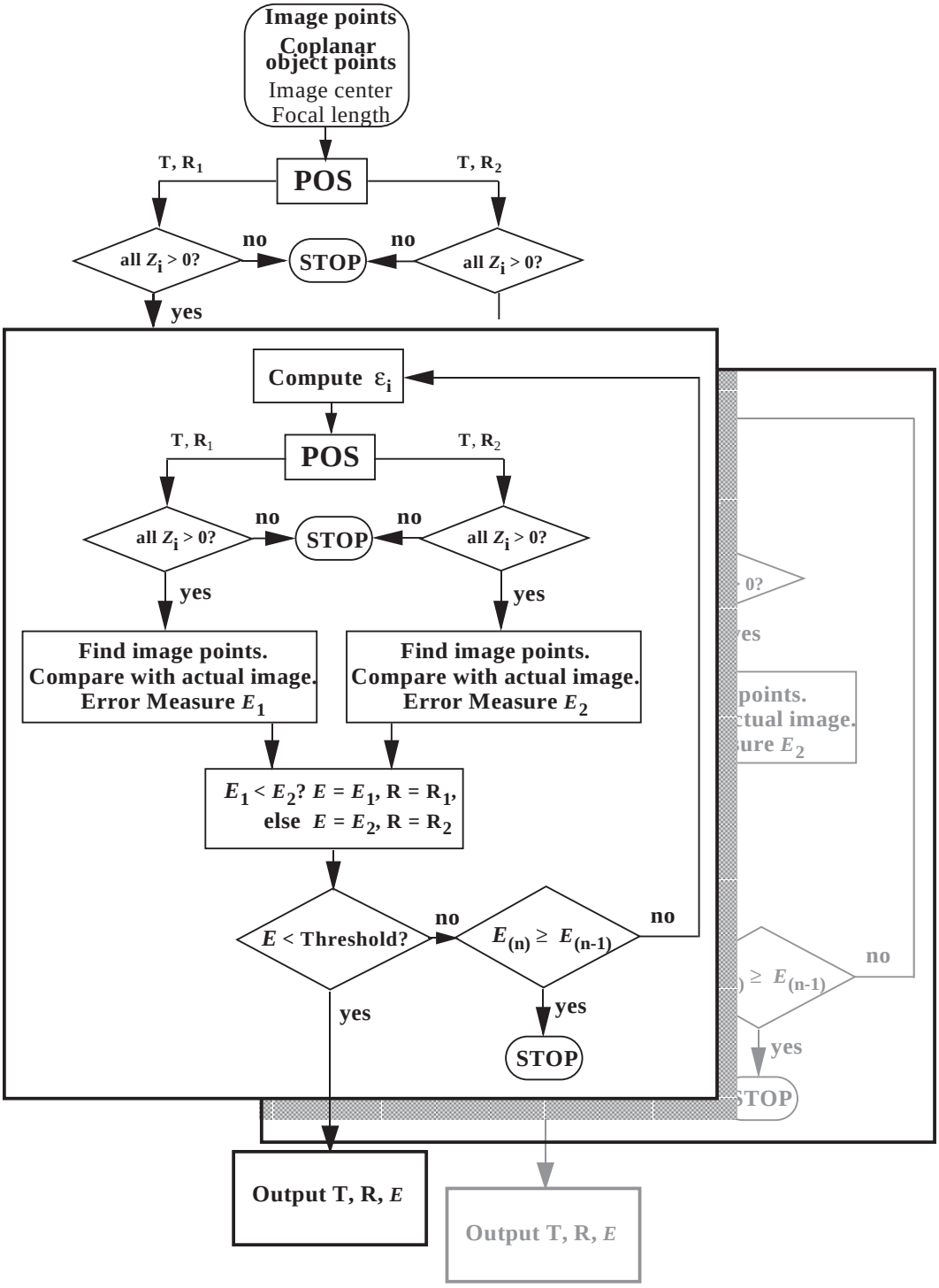

Figure 3.5: Algorithme POSIT pour points coplanaires, avec deux branches correspondant aux deux solutions; l'organigramme en arrière-plan correspond à la deuxième solution

est abandonnée si les coordonnées en z de certains points de l'objet sont négatives. Pour la seconde itération et les itérations suivantes, les calculs pour chacune des deux branches de la première itération sont similaires, et on ne montre au premier plan que le processus pour la branche de gauche. Ce processus consiste à choisir la meilleure parmi deux poses par la mesure de distance E mentionnée plus haut, et à vérifier si E est au–dessous d'un seuil prédéfini en fonction du niveau de bruit d'image. Si c'est le cas, la pose pour cette branche est affichée, ainsi que la mesure de distance correspondante E; dans le cas contraire, la pose est utilisée au pas d'itération suivant pour recalculer les termes ε_i.

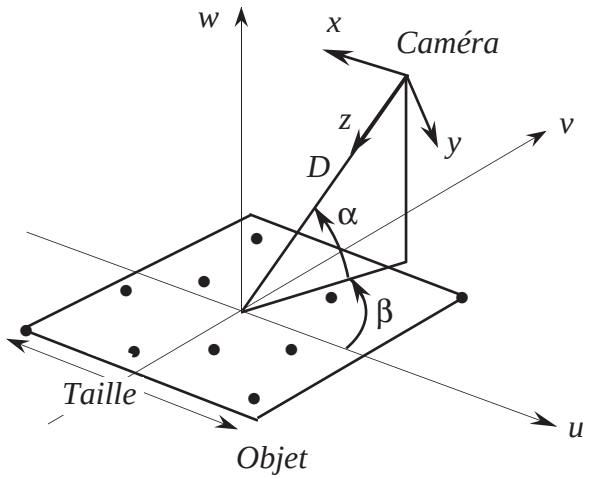

Figure 3.6: Élévation α et azimuth β de la caméra dans les expériences

3.7 Mesure de performance

Dans cette section, nous évaluons les erreurs d'orientation et de position de l'algorithme POSIT appliqué à des points coplanaires. Par une démarche similaire à celle du chapitre précédent, des images synthétiques sont créées, et les poses calculées par POSIT sont comparées avec les poses réelles qui ont servi à calculer ces images. Ici, nous faisons ces calculs pour un objet comprenant dix points coplanaires, à quatre distances de la caméra, moyennant pour chaque distance les erreurs obtenues pour 72 azimuths à intervalle régulier par rapport au plan de l'objet. Pour un objet plan, il a paru logique de définir les orientations de la caméra par rapport à l'objet en termes d'azimuth et d'élévation, et de comparer les erreurs des vues de face (élévation = 90 degrés) et rasantes (élévation = 10 degrés). Nous répétons les expériences pour trois niveaux de bruit d'images. Le calcul des images synthétiques et l'addition de bruit sont effectués de la manière décrite au chapitre précédent (sections 2.15.3 et 2.15.4). Les erreurs d'orientation et de position sont définies et calculées de la même manière.

3.7.1 Objet

L'objet comprend deux points diagonalement opposés dans un carré de 10 cm, et huit autres points à des positions aléatoires à l'intérieur du carré. L'origine O du repère $(O\mathbf{u}, O\mathbf{v}, O\mathbf{w})$ du repère de l'objet est au centre du carré, et le plan $W = 0$ est le plan du carré (figure 3.6).

3.7.2 Positions de l'objet

La caméra pointe toujours vers l'origine O de l'objet et elle est positionnée successivement à quatre distances de ce point, 2, 5, 10 et 20 fois la dimension du carré contenant les points (10 cm). Pour chacune de ces distances, on considère 17 angles d'élévation, de 10 à 90 degrés. Ces élévations sont notées α sur la figure 3.6. Pour la dernière élévation, la caméra fait face au carré de l'objet. Toutes les erreurs de poses sont représentées en fonction de ces élévations dans les diagrammes. Cette représentation est différente de celle du chapitre précédent, parce qu'ici il est intéressant de comparer les erreurs à différents angles de la caméra, par exemple

entre vues rasantes et vues de face.

La figure 3.6 illustre aussi la définition de l'azimuth β de la caméra. Notre but a été d'obtenir si possible des résultats indépendants de la distribution des points sur le plan; les variations des résultats pour la même élévation et différents azimuths reflètent l'hétérogénéité de la distribution des points; nous prenons donc la moyenne des erreurs de pose obtenues pour 72 azimuths β en incréments de 5 degrés. Ces erreurs moyennes sont représentées avec des barres d'écart-type pour les 17 angles d'élévation et les quatre distances de caméra, pour trois niveaux de bruit.

3.7.3 Erreurs moyennes de pose

Les diagrammes de la figure 3.7 et de la figure 3.8 présentent les erreurs d'orientation et de position obtenues en moyennant les erreurs pour 72 angles d'azimuth à intervalles réguliers autour de l'objet (5 degrés).

On note sur la figure 3.7 que pour les distances jusqu'à 10 fois la dimension de l'objet et les angles d'élévation inférieurs à 35 degrés, les erreurs d'orientation sont souvent moins de 3 degrés, même pour le niveau de bruit le plus élevé. Mais des erreurs plus élevées (10 degrés) se produisent quand la caméra fait face au plan de l'objet. En effet toutes les rotations autour d'axes dans le plan de l'objet déplacent les points de l'objet dans des directions proches des directions des lignes de vue, et sont donc difficilement détectées parce qu'elles produisent peu de changements dans l'image. Par contre, en vue rasante, la plupart des rotations déplacent les points dans des directions presque normales aux lignes de vue et produisent de grands changements de positions des points d'image. Les erreurs représentées sont une image globale de toutes les erreurs de rotation possibles; au nadir, un seul type de rotation parmi toutes les rotations possibles est *facile* à détecter: la rotation autour d'un axe normal au plan de la caméra.

La figure 3.8 montre que les calculs de position sont moins sensibles au bruit d'image que les erreurs de rotation quand la caméra fait face à l'objet, avec des erreurs toujours inférieures à 6%, même à une distance de 20 dimensions avec le niveau de bruit le plus élevé. On peut l'expliquer par le fait qu'un seul type de translation est *difficile* à détecter: la translation

perpendiculaire au plan de la caméra. Pour l'ensemble des diagrammes, le calcul de position paraît plus précis et moins sensible au bruit que le calcul d'orientation.

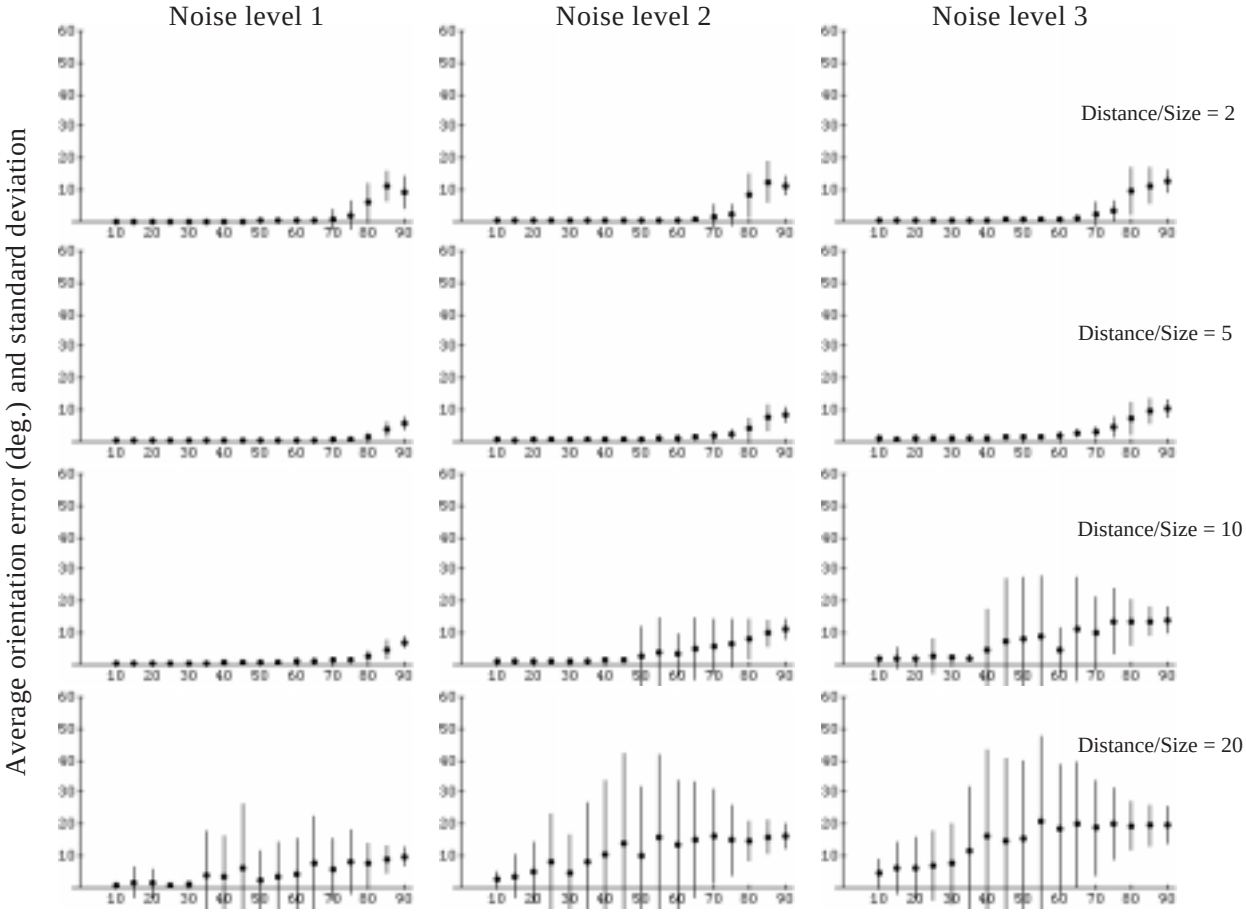

Figure 3.7: Erreurs d'orientation avec barres d'écarts-types en fonction des angles d'élévation α pour 10 points coplanaires. Ces résultats sont des moyennes pour 72 azimuths autour de l'objet.

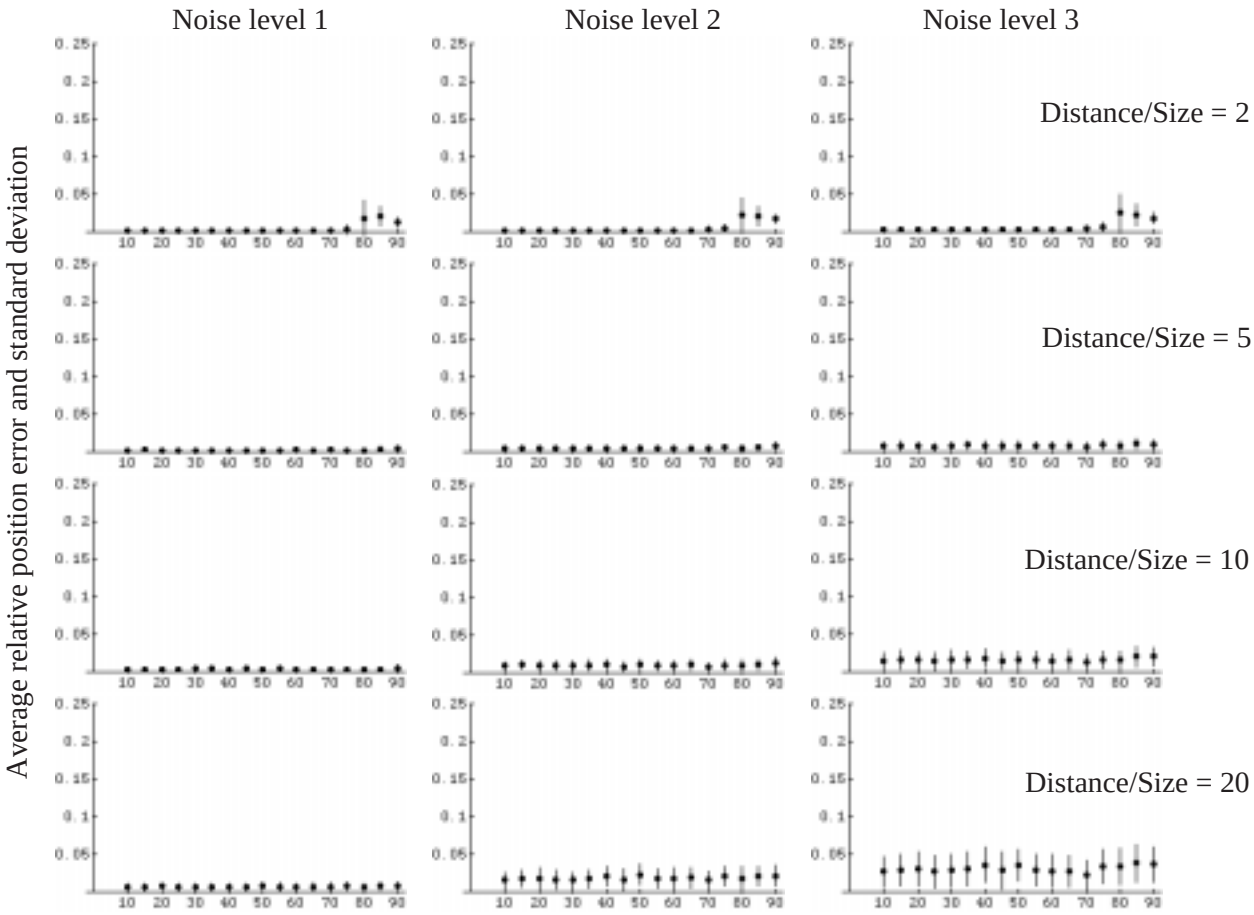

Figure 3.8: Erreurs de position avec barres d'écarts-types en fonction des angles d'élévation α pour 10 points coplanaires. Ces résultats sont des moyennes pour 72 azimuths autour de l'objet

3.7.4 Nombre de poses acceptables

Avec l'algorithme POSIT décrit ci-dessus, on peut trouver deux poses acceptables pour certaines positions de la caméra. Une pose est appelée *acceptable* si elle satisfait la définition suivante: on considère les écarts entre les points d'image réels et les projections des points caractéristiques pour cette pose; la pose est acceptable si tous ces écarts sont inférieurs à l'amplitude du bruit d'image (cette amplitude est 0,5 pixel pour le niveau de bruit 1, 1,5 pixel pour le niveau 2, 2,5 pixels pour le niveau 3). En effet, en pratique, si les écarts sont inférieurs au bruit, on ne peut pas savoir si ces écarts sont dus au fait que les points d'image ont été détectés avec un décalage à cause du bruit d'image, ou bien au fait que les projections sont décalées à cause d'une erreur de pose. On donne le bénéfice du doute à la pose, et on juge la pose acceptable. Dans tous les cas, on trouve au moins une pose acceptable; dans certaines configurations, on a des chances de trouver deux poses acceptables.

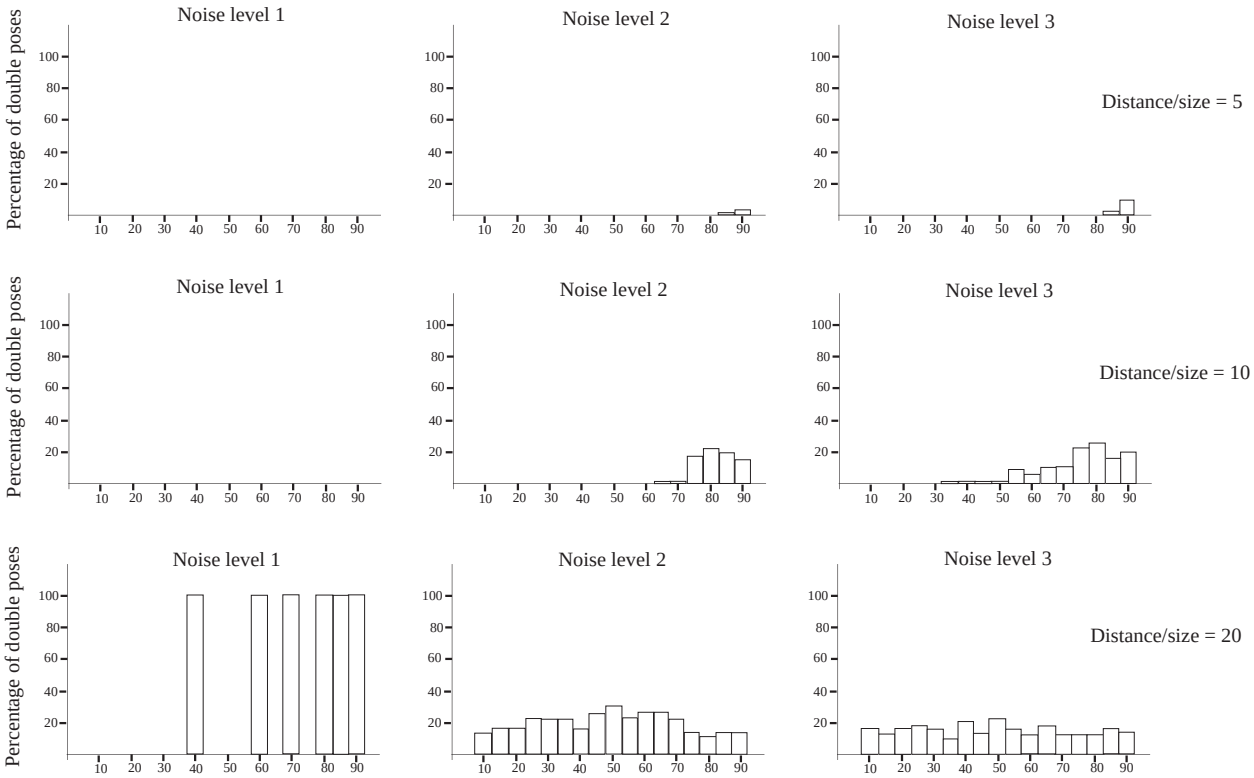

Figure 3.9: Probabilités d'obtenir deux poses acceptables en fonction des angles d'élévation α, pour 10 points coplanaires.

Les diagrammes de la figure 3.9 présentent les résultats sous forme de pourcentages de doubles poses pour les angles d'élévation de la caméra entre 10 et 90 degrés, et peuvent être interprétés comme des probabilités de trouver deux poses acceptables au lieu d'une. Le protocole de calcul de ces pourcentages est le suivant: pour chacune des élévations indiquées en abscisse, on génère 72 images bruitées de l'objet en perturbant les positions nominales des points d'image par des écarts horizontaux et verticaux créés par un générateur de nombres aléatoires à distribution uniforme, dans l'intervalle spécifié par le niveau de bruit indiqué en haut de la figure. Pour chacune des images, on calcule les poses par POSIT; on compte le nombre de fois pour lesquelles on obtient deux poses acceptables au lieu d'une seule, et on divise ce nombre par le nombre total d'expériences (72).

Un seul angle d'azimuth est considéré. Les trois colonnes de diagrammes correspondent aux niveaux de bruit 1, 2, 3, et les trois lignes correspondent aux distances égales à 5, 10 et 20 fois la dimension du carré contenant les points caractéristiques. On n'a pas représenté la distance la plus faible des diagrammes précédents (2 fois la dimension de l'objet); pour cette distance, tous les pourcentages sont zéro, c'est-à-dire que POSIT ne fournit qu'une seule pose acceptable à cette distance pour tous les angles d'élévation et les trois niveaux de bruit.

On note qu'au niveau de bruit 1, les barres de pourcentage sont soit 0, soit 100 %. En effet, le niveau de bruit 1 est dû uniquement à la discrétisation des images en pixels, et les 72 images utilisées pour chaque élévation sont donc identiques. Le fait que les résultats puissent sauter de 0 à 100 % puis de 100 % à 0 alors que l'élévation est augmentée par incréments égaux peut sembler étrange. La raison de ces résultats aléatoires semble être que pour une pose donnée, les projections des points peuvent être décalées par l'erreur de pose dans la même direction que l'erreur de discrétisation, auquel cas la pose est acceptable, ou dans une direction opposée, auquel cas la pose est rejetée. De même, pour les images aléatoires des niveaux 2 et 3, on n'obtient jamais 100 % de doubles poses; il y a toujours des expériences pour lesquelles les erreurs de pose et d'image décalent les points dans des directions opposées.

On voit sur les diagrammes qu'on a plus de chances de trouver deux poses quand le rapport de la distance de la caméra à l'objet divisée par la projection de l'objet sur l'axe optique est élevé. En effet, quand ce critère s'applique, la projection orthographique à l'échelle est une bonne approximation de la projection perspective, et on sait qu'avec cette

approximation on trouve toujours deux poses quand les points sont coplanaires. Ce critère est vérifié lorsque la caméra fait face à l'objet, ou lorsque l'objet est loin de la caméra, ou avec la combinaison d'une distance moyenne et une caméra à une incidence non rasante. On ne trouve deux poses à la distance la plus faible que lorsque la caméra fait face à l'objet; pour la distance intermédiaire, on commence à trouver deux poses pour les incidences intermédiaires de la caméra; pour la distance la plus grande, la probabilité de trouver deux poses est pratiquement indépendante de l'angle de la caméra.

Nous avons aussi effectué des séries d'expériences avec un objet comprenant 4 points coplanaires seulement (figure 3.10). On observe que les erreurs de pose sont plus élevées aux niveaux de bruit 2 et 3. Il semble qu'avec un plus grand nombre de points, l'information de pose dans l'image devient très redondante et permet donc de reconstruire une pose plus exacte grâce à l'effet de la matrice pseudo–inverse qui compense le bruit d'image. D'autre part, on observe qu'avec seulement 4 points, les probabilités d'avoir deux poses sont beaucoup plus élevées (plus de 90 % aux positions pour lesquelles on trouve seulement 20 % avec 10 points). Une explication possible est que quand une deuxième pose existe pour 10 points, elle est aussi calculée de manière plus exacte, et a donc moins de chance de se trouver par hasard dans l'intervalle acceptable que pour 4 points. Toutefois, il nous est difficile d'accepter que cet effet seul puisse produire une telle différence de probabilités, et d'autres mécanismes sont à découvrir. C'est un des avantages de ces diagrammes de mettre en lumière des propriétés qui auraient été difficiles à découvrir par une étude analytique.

Enfin nous avons effectué des simulations avec trois points caractéristiques, pour des poses dans lesquelles il y clairement 4 poses possibles. Nous avons pour cela utilisé l'exemple fourni par Fischler et Bolles [19]. Toutefois, l'algorithme fournit une seule pose dans ce cas. Tout se passe comme si une des poses est un point fixe d'attraction pour l'algorithme itératif, et les autres poses sont des points fixes de répulsion.

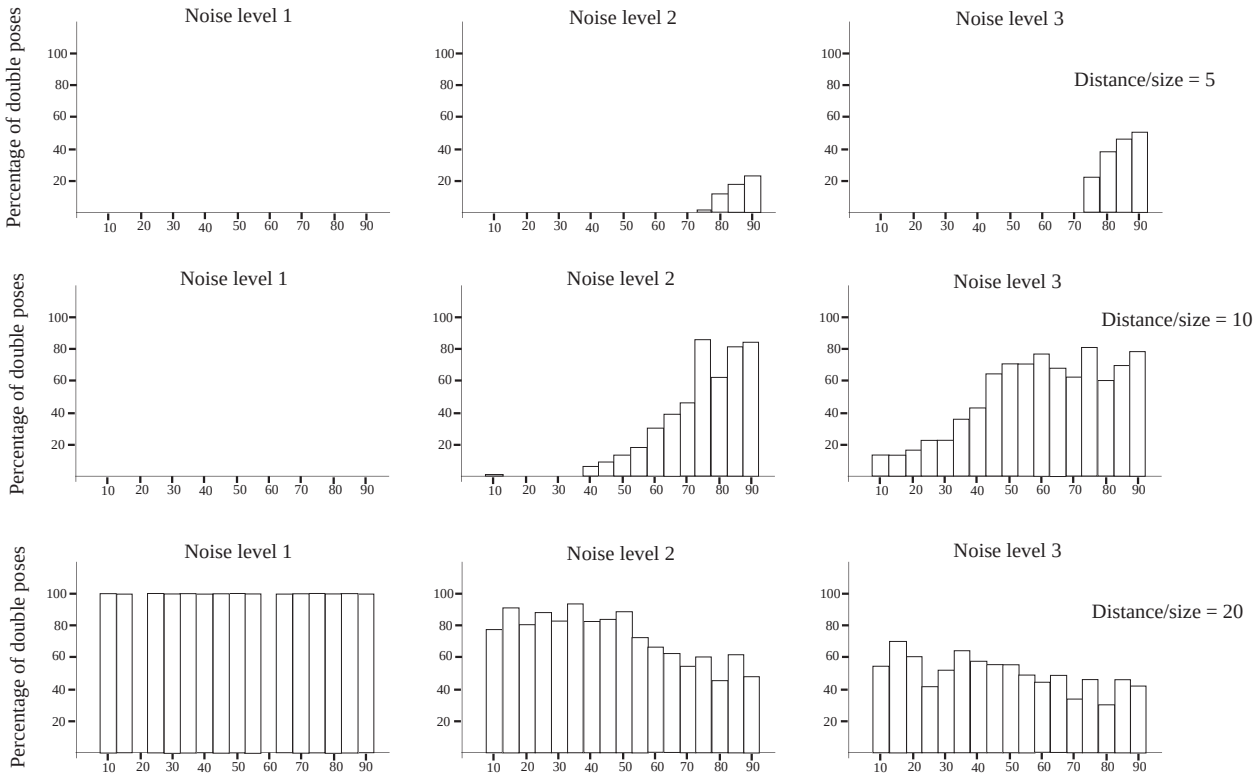

Figure 3.10: Probabilités d'obtenir deux poses acceptables en fonction des angles d'élévation α, pour 4 points coplanaires.

3.8 En résumé

Nous avons présenté une méthode itérative pour le calcul de la pose d'un objet dont les points caractéristiques sont coplanaires. Cette méthode est une extension de l'algorithme POSIT présenté au chapitre précédent pour le cas des points non coplanaires. Cette méthode est souple en ce sens qu'elle peut être utilisée aussi bien avec 4 qu'avec 100 points. Dans tous ces cas, les calculs sont économiques et ne nécessitent pas d'inversions repétées de matrices durant les itérations. Contrairement aux solutions analytiques, cette méthode est capable de déterminer la présence de deux poses acceptables lorsque l'objet est loin de la caméra. Cette méthode paraît utile en photogrammétrie. On se reportera à [42] pour plus de détails.

Pour une souris 3D, une configuration coplanaire de sources a certains avantages; par exemple, il n'y a pas de risques de superposition d'images de sources à moins que le plan des sources contienne le centre de projection. Mais lorsque le plan fait presque face à la caméra, le choix entre les deux poses est difficile, et les erreurs d'orientation sont plus élevées qu'avec une configuration non coplanaire; de plus les calculs sont plus compliqués. Dans notre mise en application pour une souris 3D, nous avons préféré utiliser des configurations non coplanaires de sources.

Chapitre 4

Mise en Correspondance Automatique des Points d'Objet et d'Image

Dans ce chapitre, nous examinons le problème général de mise en correspondance entre les points caractéristiques d'un objet et les points d'image. Nous montrons que chaque correspondance possible entre un point caractéristique et un point d'image contraint l'extrémité d'un vecteur **I**, proportionnel à la première ligne de la matrice de pose de l'objet, à appartenir à une "tranche d'espace" perpendiculaire au *vecteur caractéristique* joignant l'origine du repère de l'objet au point caractéristique, à une position définie par l'abscisse du point d'image. L'épaisseur de cette tranche d'espace est définie en fonction du bruit d'image. Similairement, le vecteur **J** proportionnel à la deuxième ligne de la matrice de pose de l'objet appartient à une tranche perpendiculaire au même vecteur, mais à une position définie par l'ordonnée du point d'image. La correspondance maximale entre l'objet et l'image est celle pour laquelle le plus grand nombre de tranches définies par les abscisses des points d'image se coupent entre elles en un volume d'espace réduit R_x définissant la position de **I**, et pour laquelle les tranches correspondant aux ordonnées des mêmes points se coupent entre elles

en un volume réduit R_y définissant la position de **J**. Nous cherchons les volumes R_x et R_y correspondant à ces intersections maximales en divisant l'espace en deux arbres binaires T_1 et T_2 dont les branches sont des boîtes de plus en plus petites, jusqu'à ce qu'on trouve une boîte contenue dans R_x et une boîte dans R_y. On effectue les deux recherches simultanément dans les deux arbres de manière à élaguer les branches qui ne peuvent contenir respectivement les extrémités de vecteurs **I** et **J** perpendiculaires et de même norme. Nous montrons que d'autres conditions permettent de réduire considérablement le nombre de branches à explorer, et que la plus grande partie du travail d'exploration peut être effectuée dans une seule dimension, le long des vecteurs caractéristiques.

4.1 Introduction

Nous présentons ici une nouvelle méthode pour résoudre simultanément le problème de correspondance et le problème de calcul de pose. Nous montrons que cette méthode est malgré tout généralement coûteuse pour des applications en temps réel. Cette conclusion justifie les méthodes plus pragmatiques présentées dans les chapitres suivants. Nous discutons les conditions dans lesquelles cette méthode générale pourrait être appliquée avec profit.

Considérons l'exemple qui nous a conduit à développer ces techniques: On aimerait construire un pointeur qui comprend une sphère sur laquelle sont réparties dix sources lumineuses. La répartition des sources sur la sphère est connue. La sphère est opaque, si bien qu'on ne voit dans chaque position de la sphère qu'une partie des sources. On obtient en général quatre, cinq ou six taches claires dans l'image, mais il est possible que des sources parasites n'appartenant pas au pointeur créent des taches supplémentaires. Les positions des taches dans l'image sont détectées avec une certaine erreur, et on peut garantir une borne supérieure de cette erreur. Par exemple on peut souvent garantir que les centres des taches sont en fait à l'intérieur d'un carré de trois ou quatre pixels. La question qu'on se pose est la suivante: Est-il possible de trouver la pose de la sphère, et en même temps de dire quelles sont les sources de lumière du pointeur qui sont visibles dans l'image? La méthode présentée ici résout ce problème. Au lieu d'utiliser des outils d'intelligence artificielle pour raisonner sur les différentes interprétations possibles de l'image, cette méthode utilise des outils de

géométrie algorithmique.

En fait, le problème qu'on vient de poser est le problème classique de reconnaissance d'objet basée sur un modèle, qui a fait l'objet de nombreux travaux de recherche ces dernières années, surtout au MIT. Nous formulons ce problème avec les techniques introduites par Baird [2] et étendues par Cass [8], Breuel [5, 6], et d'autres chercheurs [24, 25]. L'approche consiste à apparier les points du modèle et les points de l'image pour déterminer la pose de l'objet, tout en supposant qu'il y a des incertitudes introduites en détectant ces points, que certains points du modèle n'ont pas d'image, et que certains points de l'image sont des points parasites. Les points d'image réels sont quelque part dans des régions couvrant les positions auxquelles ils sont détectés. Le problème posé avec ce type de modélisation de l'incertitude est parfois appelé *problème de reconnaissance avec erreurs bornées* (bounded error recognition problem). Des résultats ont surtout été obtenus jusqu'à présent pour la mise en correspondance de deux images et pour la reconnaissance d'objets plans. Une des difficultés pour l'extension de ces résultats à des objets généraux et non plats provenait du fait que la recherche de solutions dans le cas général était effectuée dans un espace de transformation à 8 dimensions [8]. La méthode proposée dans ce chapitre décompose la recherche de solutions en un problème d'explorations dans une seule dimension utilisant des *arbres de segments*, le long de lignes géométriques définies dans le modèle de l'objet.

Nous partons des équations 2.2 et 2.3, et modifions ces équations en faisant passer au premier membre les coordonnées x_0, y_0 de l'image de l'origine M_0. On considère ces coordonnées comme inconnues, ce qui est nécessaire puisqu'on ne sait pas où est l'image de M_0 ou si M_0 a même une image. Nous obtenons des contraintes linéaires, dans un espace homogène 4D, pour deux vecteurs $\mathbf{I}$ et $\mathbf{J}$ proportionnels à la première et deuxième rangée de la matrice homogène de transformation. Ces contraintes linéaires représentent des *tranches* d'espace qui sont perpendiculaires aux *vecteurs caractéristiques* joignant l'origine du repère de l'objet aux points caractéristiques de l'objet, en des points qui dépendent des coordonnées des projections de ces points caractéristiques. Les régions de l'espace pour lesquelles le plus grand nombre de tranches s'intersectent correspondent aux appariements maximaux entre les points caractéristiques et leurs images. Pour trouver ces régions, nous adaptons la recherche par arbre binaire proposée par Breuel [6] pour ce type de problème; toutefois, dans notre

formulation, la recherche peut être décomposée en recherches le long d'une seule dimension utilisant des arbres de segments le long des vecteurs caractéristiques. Des recherches simultanées sont entreprises pour les régions contenant les extrémités de **I** et **J**, et sont élaguées par les contraintes mutuelles qui résultent du fait que **I** et **J** appartiennent à des tranches correspondant aux mêmes points d'image. D'autres critères pour élaguer les arbres utilisent le fait que les trois premières coordonnées de **I** et **J** définissent des vecteurs qui sont perpendiculaires et de modules égaux.

Quand le système poursuit un objet dans son mouvement, les termes non linéaires ε_i des équations peuvent être évalués pour chaque image à partir de la pose de l'objet obtenue à l'image précédente, et l'espace de la recherche peut être réduit parce que la position et l'étendue de cet espace peuvent être déduites par des techniques de prédiction et la connaissance des limites de déplacements admissibles. Cette méthode permet de trouver les poses successives sans se préoccuper de la disparition de points caractéristiques due aux rotations de l'objet.

4.2 Notations

La figure 4.1 est une version simplifiée de la figure 2.1. Comme dans les chapitres précédents, les coordonnées (U_i, V_i, W_i) des points M_i dans ce repère sont connues, ainsi que les coordonnées (x_i, y_i) de chacun des points d'image m_i. Alors que dans les chapitres précédents on supposait les correspondances entre M_i et m_i connues, dans le problème de reconnaissance examiné ici, un des buts est d'être capable de pouvoir dire que m_i est bien l'image de M_i, ce qui n'est pas évident puisque la pose de l'objet est inconnue. Autrement dit, nous devons trouver les appariements entre les points d'image et les points caractéristiques. On suppose aussi que certains points M_i ne sont pas visibles, et que certains points d'image m_j ne correspondent à aucun point M_j. Enfin, on suppose que l'origine M_0 n'est généralement pas un point caractéristique, et n'est pas visible dans l'image.

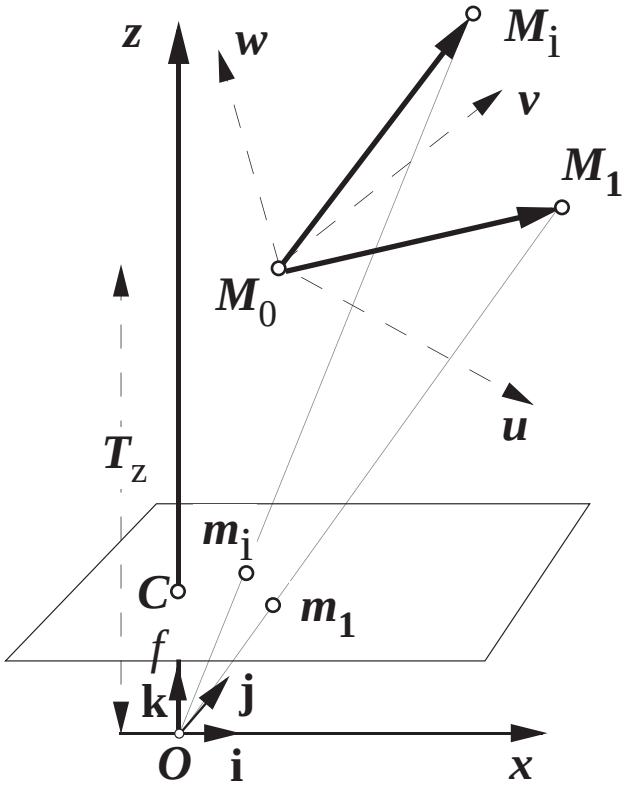

Figure 4.1: Projections perspectives m_i pour les points M_i de l'objet.

4.3 Modifications des équations fondamentales

Nous remarquons que l'équation 2.2 peut être écrite

$$[\mathbf{M}_0\mathbf{M}_i, 1] \cdot \frac{f}{Z_0}[\mathbf{i}, \frac{Z_0}{f}x_0] = x_i(1+\varepsilon_i). \tag{4.1}$$

Dans cette équation, la notation $[\mathbf{V}, a]$ est utilisée pour représenter un vecteur à quatre coordonnées dont les trois premières coordonnées sont les coordonnées d'un vecteur 3D, $\mathbf{V}$, et dont la quatrième coordonnée est égale au scalaire a. On peut vérifier dans l'équation ci-dessus que le quatrième terme du produit scalaire redonne bien le terme x_0 qui était au second membre dans l'équation 2.2. Dans la suite, on utilise les caractères gras pour noter à la fois les vecteurs 4D et 3D, et on spécifiera quand les vecteurs notés ainsi sont en fait des vecteurs 3D.

Comme on l'a vu au chapitre 2, la quantité $\frac{Z_0}{f}x_0$ est la coordonnée T_x du vecteur de translation $\mathbf{T}$ de l'objet, et les 3 coordonnées i_u, i_v, i_w du vecteur $\mathbf{i}$ dans le repère de l'objet sont les trois éléments de la première ligne de la matrice de rotation. On se rappelle aussi que la rotation et la translation d'un objet sont souvent exprimées en une seule matrice $\mathbf{P}$ que nous appellerons *matrice de pose* dans la suite. Cette matrice s'écrit

$$\mathbf{P} = \begin{bmatrix} i_u & i_v & i_w & T_x \\ j_u & j_v & j_w & T_y \\ k_u & k_v & k_w & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Les vecteurs lignes de la matrice $\mathbf{P}$ sont notés $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ et $\mathbf{P}_4$. Avec ces notations, l'équation 4.1 s'écrit

$$\mathbf{M}_0\mathbf{M}_i \cdot \frac{f}{Z_0}\mathbf{P}_1 = x_i(1+\varepsilon_i)$$

ou encore

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x_i' \tag{4.3}$$

On transforme de même l'équation 2.3:

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{J} = y_i' \tag{4.4}$$

Dans ces équations, on a introduit les notations supplémentaires

$$\mathbf{I} = \frac{f}{T_z}\mathbf{P}_1, \quad \mathbf{J} = \frac{f}{T_z}\mathbf{P}_2, \tag{4.5}$$

$$x'_i = x_i(1+\varepsilon_i), \quad y'_i = y_i(1+\varepsilon_i), \tag{4.6}$$

avec

$$\varepsilon_i = \mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_3/T_z - 1 \tag{4.7}$$

et on rappelle que $\mathbf{M}_0\mathbf{M}_i$ est maintenant un vecteur homogène dont la quatrième dimension est égale à 1.

Si on sait calculer $\mathbf{I}$ et $\mathbf{J}$ à partir de ces équations, il est facile ensuite d'obtenir la matrice de pose $\mathbf{P}$: la quantité T_z est obtenue en remarquant que les trois premières coordonnées de $\mathbf{I}$ et $\mathbf{J}$ définissent des vecteurs 3D dont les normes sont égales à f/T_z. On peut ainsi obtenir T_z et les deux premières lignes $\mathbf{P}_1$ et $\mathbf{P}_2$ de la matrice de pose $\mathbf{P}$; la troisième ligne $\mathbf{P}_3$ est obtenue en utilisant le fait que le vecteur $\mathbf{k}$ est le produit vectoriel des deux vecteurs $\mathbf{i}$ et $\mathbf{j}$.

Au lieu de passer par la démonstration géométrique du chapitre 2, on peut vérifier analytiquement que les équations 4.3 et 4.4 expriment bien la relation de projection perspective entre m_i et M_i. On considère pour cela les coordonnées (X_i, Y_i, Z_i) des vecteurs $\mathbf{M}_0\mathbf{M}_i$ *dans le repère de la caméra*, dans le but de démontrer que ces équations sont correctes. On remarque ensuite que le produit scalaire $\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_1$ est l'opération effectuée quand on veut passer du repère de l'objet au repère de la caméra: on obtient avec ce produit scalaire la coordonnée en x, X_i, de M_i dans le repère de la caméra. De même, si on effectue le produit scalaire $\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_3$, on obtient Z_i. Par conséquent, par l'équation 4.7, la quantité $(1+\varepsilon_i)$ utilisée pour définir x'_i et y'_i est en fait égale à Z_i/T_z. De plus, en projection perspective, nous avons la relation $x_i = fX_i/Z_i$. Lorsqu'on utilise ces relations dans la première équation, on obtient une identité. On vérifie la deuxième équation de la même manière.

Si les correspondances entre les points caractéristiques et les points d'image ont été établies par une autre méthode, les équations 4.3 et 4.4 peuvent être utilisées pour trouver la matrice de pose, par une variante de l'algorithme POSIT présentée en Annexe B. Mais dans les sections qui suivent, le but est d'utiliser ces équations pour découvrir les correspondances.

4.4 Contraintes géométriques pour I et J

La discussion qui suit montre que les solutions **I** et **J** sont situées à l'intérieur de petites régions polyédriques qui peuvent être identifiées dans le repère homogène 4D de l'objet.

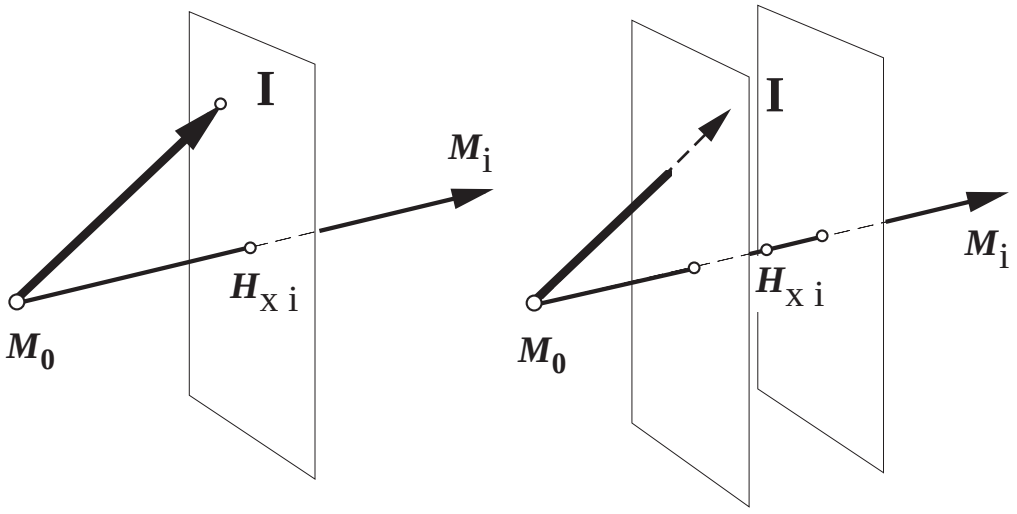

Figure 4.2: *A gauche:* En l'absence d'incertitudes, l'extrémité du vecteur **I** appartient au plan perpendiculaire à $\mathbf{M}_0\mathbf{M}_i$ au *point-x* H_{xi} situé à l'abscisse $x'_i/|\mathbf{M}_0\mathbf{M}_i|$.
A droite: A cause des incertitudes dans l'image et dans l'estimation des ε_i, on peut seulement dire que l'extrémité de **I** est contenue dans une *tranche* perpendiculaire a $\mathbf{M}_0\mathbf{M}_i$.

Nous revenons à l'interprétation géométrique des équations que nous avons utilisées aux chapitres précédents. Mais cette fois, l'espace considéré a quatre dimensions. Les équations $\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x'_i$ peuvent être interprétées comme des contraintes géométriques pour la position du vecteur **I** dans l'espace 4D par rapport aux vecteurs caractéristiques $\mathbf{M}_0\mathbf{M}_i$: Si l'origine du vecteur **I** coincide avec M_0, l'extrémité de **I** doit se projeter sur le vecteur caractéristique $\mathbf{M}_0\mathbf{M}_i$ au point H_{xi} d'abscisse $x'_i/|\mathbf{M}_0\mathbf{M}_i|$. Autrement dit, **I** doit appartenir au plan perpendiculaire à $\mathbf{M}_0\mathbf{M}_i$ en H_{xi} (figure 4.2). Dans la suite, les points H_{xi} sont appelés *points-x*. Ce sont des points construits sur les lignes des vecteurs caractéristiques $\mathbf{M}_0\mathbf{M}_i$ à partir des coordonnées en x des points d'image. De même, les points H_{yi} considérés pour construire le vecteur **J** ont pour abscisse $y'_i/|\mathbf{M}_0\mathbf{M}_i|$ le long de ces vecteurs caractéristiques et sont appelés points-y.

Dans de nombreuses situations, les termes $x'_i = x_i(1+\varepsilon_i)$ ne sont connus que de manière

approximative. Les limites des incertitudes de ces termes peuvent être calculées en ajoutant l'incertitude de détection dans l'image, e, et l'erreur faite en estimant $x_i\varepsilon_i$. Les termes ε_i sont les projections des vecteurs $\mathbf{M_0M}_i$ sur l'axe optique de la caméra, divisées par la projection T_z du vecteur OM_0 sur l'axe optique (équation 4.7). Par conséquent, R/D est une borne supérieure pour ε_i, si on appelle R le rayon de la sphère centrée en M_0 et contenant tous les points caractéristiques de l'objet, et D est une borne inférieure estimée pour la distance T_z. Des bornes plus étroites peuvent être estimées si on a une idée de la distance de l'objet. Quand on connaît une pose approchée de l'objet, les termes ε_i peuvent être estimés à partir de cette pose, et les intervalles d'incertitude peuvent être considérablement réduits.

A cause de ces incertitudes, tout ce qu'on peut dire est que l'extrémité de $\mathbf{I}$ se projette sur le vecteur caractéristique $\mathbf{M_0M}_i$ dans l'intervalle d'incertitude autour du point-x H_{xi}. Autrement dit, l'extrémité de $\mathbf{I}$ doit appartenir à la *tranche* d'espace perpendiculaire à $\mathbf{M_0M}_i$ au point H_{xi} qui a une épaisseur définie par l'intervalle d'incertitude (figure 4.2).

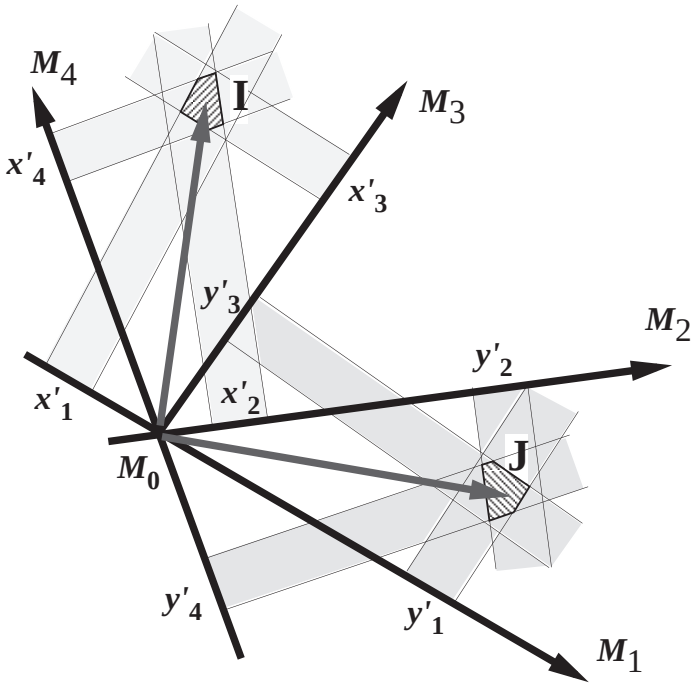

Figure 4.3: Les extrémités du vecteur $\mathbf{I}$ et du vecteur $\mathbf{J}$ se trouvent à l'intersection de tranches perpendiculaires aux vecteurs caractéristiques $\mathbf{M_0M}_i$ de l'objet. Les abscisses de ces tranches le long de ces vecteurs dépendent des coordonnées des images des points M_i. On utilise aussi le fait que les projections 3D de $\mathbf{I}$ et $\mathbf{J}$ sont perpendiculaires et de même module.

Pour que $\mathbf{I}$ soit solution d'un système de n équations 4.3 et 4.4 pour $i = 1, 2, 3, ..., n$, l'extrémité du vecteur $\mathbf{I}$ doit appartenir à la tranche S_{11} définie au point-x H_{x1} sur le vecteur caractéristique $\mathbf{M}_0\mathbf{M}_1$, la tranche S_{22} définie en H_{x2} sur $\mathbf{M}_0\mathbf{M}_2$, etc.... Par conséquent, l'extrémité de $\mathbf{I}$ doit appartenir à l'intersection de ces n tranches. Une condition nécessaire pour que cela se produise est qu'il existe une région de l'espace R_x contenue dans au moins n tranches T_{xii}. Cette condition est illustrée en deux dimensions sur la figure 4.3.

De même, $\mathbf{J}$ doit appartenir à l'intersection de n tranches T_{yii}. Une condition nécessaire pour que cela se produise est qu'il existe une région R_y de l'espace contenue dans au moins n tranches T_{yii} (figure 4.3).

De plus, les n tranches T_{xii} contenant la région R_x et les n tranches T_{yii} contenant la région R_y doivent être définies à partir des mêmes vecteurs caractéristiques $\mathbf{M}_0\mathbf{M}_i$ et des mêmes points d'image m_i. Autrement dit, les n tranches T_{yii} définies aux points-y H_{yi} calculés à partir des mêmes points $\mathbf{m}_i$ utilisés pour les tranches T_{xii} doivent se couper dans une région non vide R_y.

Finalement, les solutions $\mathbf{I}$ et $\mathbf{J}$ sont soumises à des contraintes de module et d'orientation; les trois premières coordonnées de $\mathbf{I}$ et $\mathbf{J}$ définissent deux vecteurs 3D, $\mathbf{R}_1$ et $\mathbf{R}_2$. Ces vecteurs sont proportionnels à $\mathbf{i}$ et $\mathbf{j}$ respectivement, avec les mêmes coefficients de proportionnalité f/T_z (voir section 4.3). Donc $\mathbf{R}_1$ et $\mathbf{R}_2$ doivent être orthogonaux et de même module. Par conséquent, une paire de régions R_x et R_y ne peut contenir les extrémités de $\mathbf{I}$ et $\mathbf{J}$ que si (1) les valeurs extrêmes des produits scalaires 3D entre les paires de vecteurs dont les extrémités appartiennent à chaque région sont de signes opposés, et (2) le domaine des distances de M_0 aux points de R_x chevauche le domaine des distances de M_0 aux points de R_y.

4.5 Recherche des régions solutions sans connaissance des correspondances

Dans le problème considéré, les correspondances entre les N points caractéristiques et les n' points d'image détectés sont inconnues. Considérant un point caractéristique M_i, on ne sait pas quel point parmi les points d'image m_1, m_2, m_3, etc..., est l'image de M_i. De plus, certains points M_i n'ont pas d'image, et certains points d'image sont des points parasites qui ne correspondent à aucun point caractéristique. Toutefois, nous supposons pour l'instant que le nombre n_0 de points d'image qui peuvent être appariés à des points caractéristiques *est connu* (la découverte de ce nombre est le sujet de la section suivante). Notre seul recours dans ces conditions est de considérer, pour chacun des N points caractéristiques M_i, toutes les tranches d'espace définies à partir des n' points d'image détectés. Sur chaque vecteur caractéristique $\mathbf{M_0M}_i$ on peut construire un point-x pour chaque point d'image détecté $\mathbf{m}_j$, et construire la tranche d'espace pour ce point. Toutes les tranches pour un vecteur caractéristique donné $\mathbf{M_0M}_i$ sont parallèles. Les tranches pour deux vecteurs caractéristiques différents se coupent si leurs points caractéristiques M_i ne sont pas alignés avec M_0.

La méthode proposée recherche de petites régions de l'espace R_x et R_y qui contiennent les extrémités des vecteurs $\mathbf{I}$ et $\mathbf{J}$. S'il existe bien n_0 images des points caractéristiques parmi les points d'image détectés, et si les incertitudes sont modélisées correctement, une paire de régions (R_x, R_y) *doit exister* telle que

1. R_x est contenue dans au moins n_0 tranches définies par des points-x H_{xij} placés sur les vecteurs $\mathbf{M_0M}_i$ et dépendant des coordonnées en x des points d'image m_j;

2. R_y est aussi contenue dans au moins n_0 tranches définies par des points-y H_{yij} placés sur ces mêmes vecteurs $\mathbf{M_0M}_i$ et dépendant des coordonnées en y des mêmes points d'image m_j.

La méthode de recherche de ces régions R_x et R_y s'appuie sur deux subdivisions récurrentes de l'espace (une pour rechercher $\mathbf{I}$ et une pour rechercher $\mathbf{J}$) en boîtes dont les faces sont perpendiculaires aux axes du repère de l'objet. On procède dans les deux subdivisions par

élimination des paires de boîtes qui ne satisfont pas les contraintes géométriques définies dans la section précédente. Considérant une boîte B_x et une boîte B_y, ces boîtes *ne peuvent pas* respectivement contenir les extrémités de **I** et **J** si:

1. B_x ou B_y ne sont pas coupées par n_0 tranches (dans ce cas aucun point de B_x ou B_y ne peut être contenu dans n_0 tranches);

2. B_x et B_y ne sont pas coupées par n_0 tranches construites à partir des mêmes points d'image.

3. Le domaine des distances 3D de M_0 aux points de B_x ne chevauche pas le domaine des distances 3D de M_0 aux points de B_y (dans ce cas les deux boîtes ne peuvent pas contenir les extrémités de **I** et **J** à des distances égales de M_0).

4. Les valeurs extrêmes des produits scalaires 3D entre paires de vecteurs dont les extrémités sont contenues dans chaque boîte sont de même signe (dans ce cas les deux boîtes ne peuvent pas respectivement contenir les extrémités de deux vecteurs perpendiculaires **I** et **J**).

Il doit exister une paire de boîtes contenues respectivement dans la paire de régions (R_x, R_y) qui ne peut pas être éliminée par les critères ci-dessus, et il est donc possible de reconnaître dans l'image les n_0 points qui contribuent à ces boîtes. Nous découvrons ces boîtes par une division récurrente de l'espace pour trouver une boîte B_x, et simultanément une division récurrente de l'espace pour trouver une boîte B_y. Nous explorons simultanément deux arbres binaires, cherchant en profondeur d'abord (depth-first), appariant les branches de chacun des deux arbres et élaguant les branches appariées en utilisant les critères d'exclusion ci-dessus.

Pour obtenir ensuite une vérification supplémentaire de la mise en correspondance des n_0 points (obtenant aussi une pose plus précise), on peut alors recalculer les termes ε_i à partir de la matrice de pose, utilisant les expressions données par les équations 4.3 et 4.3. Les termes $x'_i = x_i(1+\varepsilon_i)$ et $y'_i = y_i(1+\varepsilon_i)$ sont recalculés, ainsi que des intervalles d'incertitude réduits. Ces coordonnées et ces intervalles corrigés définissent des tranches plus fines à des positions

plus correctes, qui produisent des régions R_x et R_y plus petites, des correspondances moins ambiguës entre les appariements possibles, et une pose plus précise.

4.6 Recherche des meilleures boîtes

Les explications jusqu'à présent ont porté sur la recherche des boîtes B_x et B_y à l'intersection d'au moins n_0 tranches. On aimerait en fait trouver les boîtes à l'intersection du plus grand nombre possible de tranches, parce que ces boîtes correspondent au nombre maximal d'appariements entre points d'image et points caractéristiques de l'objet. Nous démarrons donc la recherche avec $n_0 = n'$, le nombre total de points détectés dans l'image. Généralement, certains points de l'image ne correspondent à aucun point de l'objet, et ce nombre est supérieur au nombre actuel de correspondances. La recherche dans ce cas est sans succès. On décrémente alors n_0 jusqu'à ce qu'une recherche réussisse.

4.7 Recherche binaire des boîtes contenues dans les régions R_x et R_y

Puisque les régions sont définies à l'intersection de tranches d'espace, on pourrait penser à utiliser une approche de type transformée de Hough pour trouver les régions correspondant au maximum d'intersections de tranches. Toutefois Breuel [6] montre dans un contexte similaire qu'une telle approche nécessite pour ce type de problème une grande quantité de mémoire. Il montre qu'une recherche par arbre binaire est mieux adaptée. Pour la recherche d'une seule région, par exemple R_x, l'approche consisterait à démarrer avec une boîte dont les faces sont parallèles aux axes, assez grande pour contenir avec certitude la région en question, et diviser de manière récurrente la boîte en deux boîtes égales. A la profondeur 1, le plan utilisé pour diviser la boîte est perpendiculaire à l'axe des x de l'espace 4D; à la profondeur 2, le plan est perpendiculaire à l'axe des y; à la profondeur 3 le plan est perpendiculaire à l'axe des z, et à la profondeur 4 à l'axe des k, le long de la quatrième dimension de l'espace. A la profondeur 5 on retourne à une division perpendiculaire à l'axe des x, etc.... On atteint

finalement un niveau et un emplacement de l'espace où une au moins de ces boîtes est si petite qu'elle est contenue dans la région R_x et se trouve donc contenue dans n_0 tranches. Le principe de cette subdivision est illustré sur la figure 4.4.

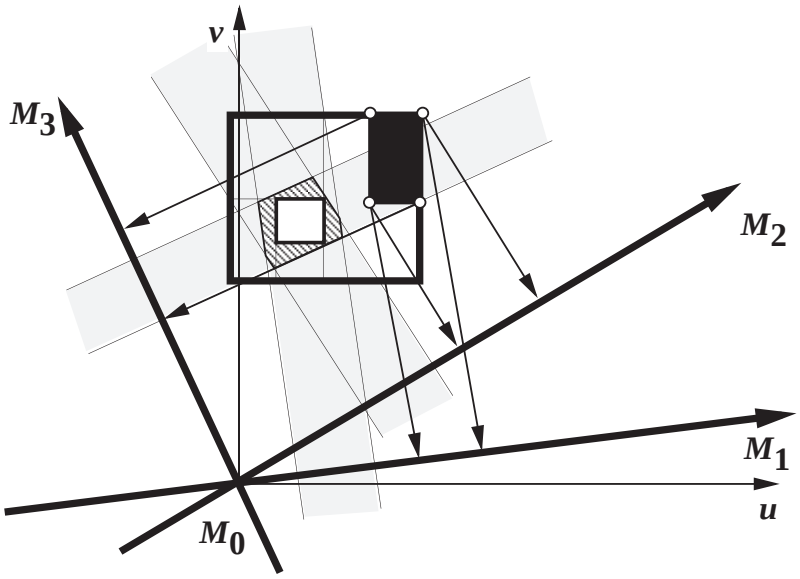

Figure 4.4: Une recherche par division de l'espace localise une boîte contenue dans une région à l'intersection de trois tranches (boîte blanche). Les boîtes qui ne sont pas *intersectées* par trois tranches peuvent être élaguées (boîte noire). Les tests d'intersection entre une boîte et des tranches peuvent être exécutées en utilisant les projections de la boîte sur chacun des vecteurs caractéristiques.

4.7.1 Recherche simultanée de deux régions

Nous débutons la recherche avec une boîte initiale A_0 assez grande pour inclure avec certitude l'extrémité de **I** et une boîte initiale B_0 assez grande pour inclure avec certitude l'extrémité de **J** (à partir des équations 4.3, 4.4 et 4.5, on peut montrer que les coordonnées de ces vecteurs peuvent être exprimées en pixels et sont inférieures aux coordonnées maximales des points d'image). La boîte A_0 est divisée en deux boîtes A_1 et A_2. La boîte B_0 est divisée en deux boîtes B_1 et B_2. A l'étape suivante on considère la paire de boîtes A_1 et B_1, et on applique à cette paire de boîtes les critères d'élimination de la section précédente. Si aucun des critères d'élimination n'est appliquable, ces deux boîtes sont elles-même divisées, et le processus est répété de manière récurrente. Si l'un des critères d'élimination s'applique, la paire de boîtes est éliminée , et une autre paire à la même profondeur, par exemple A_1 et B_2, est examinée.

Si les quatre paires possibles de boîtes ont été éliminées, on revient en arrière pour prendre une paire qui n'a pas été considérée à la profondeur précédente. Si la profondeur précédente est la profondeur de racine, on peut conclure que la recherche a échoué. La recherche réussit quand une paire de boîtes de dimension prédéfinie (quelques pixels si les coordonnées de **I** et **J** sont exprimées en pixels) survit aux critères d'élimination, et se compose donc de deux boîtes contenues dans n_0 tranches correspondantes; ce test d'inclusion n'est exécuté que lorsque les boîtes sont assez petites pour être contenues à l'intérieur des tranches.

4.7.2 Tests de l'intersection d'une boîte avec n tranches

Si une boîte n'intersecte pas n tranches, aucune subdivision de cette boîte ne peut être contenue dans n tranches, et cette boîte peut être éliminée [6]. Cette observation est un des critères d'élimination définis ci-dessus. Les tests d'intersection (et d'inclusion, voir section suivante) sont plus simples ici que dans la formulation de Breuel [6], du fait que le problème est posé sous une forme géométrique plus simple. Une boîte susceptible de contenir l'extrémité de **I** intersecte n tranches si, pour n vecteurs caractéristiques $\mathbf{M}_0\mathbf{M}_i$, la projection 1D de la boîte intersecte un intervalle d'incertitude autour d'un point-x H_{xij}.

Au lieu d'examiner les intersections entre intervalles, on augmente de chaque côté l'intervalle de la projection de la boîte par l'amplitude de l'intervalle d'incertitude, et on compte le nombre de points-x contenus dans cet intervalle. Les intervalles d'incertitude et les longueurs p_{di} de la projection d'une boîte à la profondeur d sur les vecteurs caractéristiques $\mathbf{M}_0\mathbf{M}_i$ sont précalculés (figure 4.5).

Le compte des intersections entre boîtes et tranches est incrémenté de 1 pour chaque vecteur caractéristique pour lequel on trouve au moins un point-x à l'intérieur de l'intervalle de projection (augmenté) de la boîte. Dans ce but, on trouve la liste des points-x contenus dans cet intervalle. On a au préalable trouvé la liste des points-x contenus dans l'intervalle ancêtre, à l'étape précédente de la récurrence. Chaque intervalle descendant partage une borne avec l'intervalle ancêtre. L'autre borne d'un intervalle descendant est à l'intérieur de l'intervalle ancêtre, et il suffit de localiser cette borne par rapport aux points-x de l'intervalle ancêtre. Ce résultat est obtenu par dichotomie de cette liste de points-x, qui est triée si la

liste racine avait été triée. La localisation de cette borne par rapport à la liste fournit le compte pour la nouvelle liste des points-x descendants. La liste de l'intervalle descendant de gauche a la même adresse que son ancêtre, tandis que la liste de l'intervalle descendant de droite a une adresse décalée par la position de sa borne de gauche dans la liste de l'intervalle ancêtre.

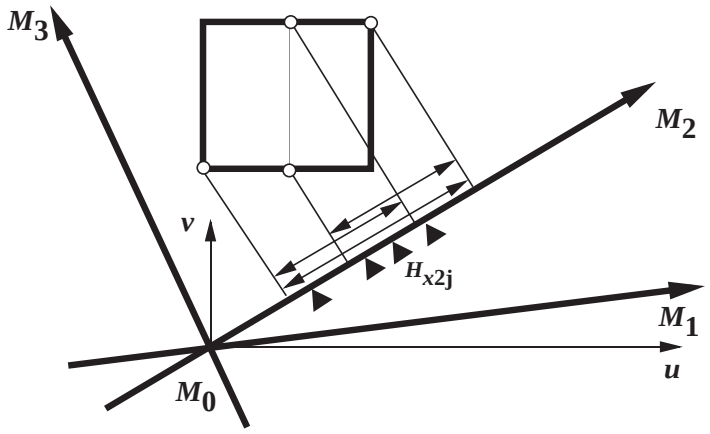

Figure 4.5: L'intervalle de projection d'une boîte descendante a une borne en commun avec l'intervalle de projection de la boîte ancêtre. Une recherche par dichotomie trouve la position de l'autre borne de cet intervalle dans la liste triée des points-x contenus dans l'intervalle ancêtre. Cette position fournit le compte des éléments et l'adresse de la liste des points-x de l'intervalle descendant.

4.7.3 Tests de l'inclusion d'une boîte dans n_0 tranches

On vérifie qu'une boîte est contenue dans n_0 tranches en vérifiant que pour au moins n_0 vecteurs caractéristiques $\mathbf{M}_0\mathbf{M}_i$, la projection 1D de cette boîte sur $\mathbf{M}_0\mathbf{M}_i$ est incluse dans l'intervalle d'incertitude autour d'un point-x H_{xij}. La profondeur pour laquelle toutes les longueurs des intervalles de projection d'une boîte à la profondeur d sur les vecteurs caractéristiques $\mathbf{M}_0\mathbf{M}_i$ sont plus petites que les longueurs u_i des intervalles d'incertitude peut être précalculée , et on ne commence à vérifier l'apparition d'inclusions qu'à partir de cette profondeur. Au lieu de vérifier si une projection de longueur p_{di} de la boîte est incluse dans un intervalle d'incertitude autour d'un point H_{xij}, il est plus simple de vérifier s'il y a un point H_{xij} contenu dans l'intervalle de longueur $(u_i - p_{di})$.

4.8 Poursuite

Dans une activité de poursuite, l'objet a été détecté dans l'image précédente, et sa pose a été calculée à partir de l'estimation de vecteurs acceptables $\mathbf{I}$ et $\mathbf{J}$. Nous proposons d'opérer la poursuite dans l'espace 4D dans lequel la recherche de $\mathbf{I}$ et $\mathbf{J}$ est effectuée. Tout d'abord, la connaissance de la pose précédente nous permet de calculer des valeurs estimées pour les termes ε_i, et de calculer les quantités $x'_i = x_i(1 + \varepsilon_i)$ et $y'_i = y_i(1 + \varepsilon_i)$ comme abscisses des points-x et points-y sur les vecteurs caractéristiques. L'erreur faite en utilisant les termes ε_i provenant de l'image précédente contribue à augmenter les intervalles d'incertitude autour de ces points et peut être estimée à partir de limites supérieures $d\theta$ et $d\mathbf{T}$ sur les angles de rotation et incréments de translation possibles entre deux images. D'autre part, le vecteur $\mathbf{I}$ devient $\mathbf{I}' = \mathbf{I} + d\mathbf{I}$ entre deux images successives, et la recherche pour l'extrémité du nouveau vecteur $\mathbf{I}'$ peut être limitée à une boîte centrée autour de l'extrémité de $\mathbf{I}$ et de dimension supérieure à $2|d\mathbf{I}|$; cette quantité peut aussi être calculée a partir de $d\theta$ et $d\mathbf{T}$. Des techniques de prédiction peuvent aussi être appliquées pour $\mathbf{I}'$ et l'incertitude sur $\mathbf{I}'$ pour réduire la dimension de la boîte dans laquelle la recherche est effectuée. Les mêmes opérations sont appliquées à $\mathbf{J}'$.

4.9 En résumé

Nous avons introduit de nouvelles équations pour exprimer la relation entre les deux premières lignes de la matrice 4×4 de pose de l'objet, les points caractéristiques d'un objet, et les points d'image obtenus par projection perspective. Ces équations groupent les termes non linéaires de la transformation perspective dans le second membre avec les coordonnées des points d'image. L'incertitude sur les estimations de ces termes non linéaires peut être modelée comme incertitude ajoutée à l'incertitude d'image. Nous obtenons des contraintes linéaires pour deux vecteurs 4D proportionnels à la première et deuxième lignes de la matrice homogène de la pose de l'objet. Ces contraintes linéaires représentent des tranches de l'espace 4D qui sont perpendiculaires aux vecteurs caractéristiques (joignant l'origine de l'objet aux points caractéristiques) en des points dépendant des coordonnées des points d'images. Les

régions d'espace où le plus grand nombre de tranches s'intersectent localisent les vecteurs **I** et **J** et correspondent aux appariements maximaux entre points caractéristiques et points d'images. Deux arbres binaires sont explorés simultanément pour la recherche de ces régions correspondant à **I** et **J**, et sont élagués par l'utilisation de contraintes mutuelles résultant du fait que **I** et **J** appartiennent respectivement à des tranches correspondant aux coordonnées x et y des mêmes points d'image. Les autres critères d'élagage utilisent le fait que les trois premières coordonnées de **I** et **J** définissent des vecteurs perpendiculaires et de même modules en 3D. La plus grande partie de la recherche consiste en une recherche 1D le long des vecteurs caractéristiques utilisant des arbres de segments.

Nous espérions appliquer cette méthode à un pointeur comprenant une dizaine de sources réparties de manière aléatoire sur une sphère opaque, mais le temps de calcul (plusieurs secondes) nous en a dissuadé. Nous avons donc dû revenir pour notre système à des méthodes plus pragmatiques de mise en correspondance (chapitre 8). Nous travaillons à présent sur l'application de cette méthode au problème de mise en correspondance automatique de points caractéristiques d'une image aérienne avec un modèle du site (voir [37] pour une différente approche). Là encore, les temps de calculs sont prohibitifs: plusieurs heures pour trouver la meilleure correspondance entre les 30 points du modèle et 200 coins de bâtiments de l'image. Il est possible toutefois que la méthode trouve son utilité dans des applications où elle peut être combinée avec d'autres méthodes capables de réduire les dimensions de l'espace de recherche.

Chapitre 5

Les Ingrédients de la Réalité Virtuelle

La réalité virtuelle est une technologie qui permet à l'opérateur de faire l'expérience de mondes tridimensionnels générés par l'ordinateur. Un traqueur[1] est généralement utilisé pour fournir à l'ordinateur la position et l'orientation de la main de l'opérateur, ou d'un pointeur tenu dans sa main. Pour permettre une expérience visuelle plus réaliste, un deuxième traqueur est nécessaire pour la pose de la tête de l'opérateur. Nous faisons la distinction entre réalité virtuelle "immersive" et réalité virtuelle "de bureau"; les masques de visualisation actuels créent une immersion complète dans le monde virtuel, mais sont encombrants et de faible résolution; de nombreuses applications sont mieux servies par les techniques 3D utilisant une console de visualisation fixe de haute résolution. Nous donnons quelques exemples d'applications dans les deux catégories. La détection du mouvement des doigts ne paraît pas nécessaire dans la plupart des applications; il est souvent préférable de représenter l'outil manipulé par la main plutôt que la main elle-même dans la scène synthétique. Nous discutons des modes d'interaction possibles entre le monde virtuel et l'opérateur, en soulignant

[1]Nous utilisons le terme *traqueur*, parfois appliqué à un radar de poursuite, pour tous les types de dispositifs qui permettent de continuellement connaître la position et l'orientation d'un objet dans l'espace.

l'importance des *métaphores* qui facilitent l'apprentissage et la construction d'un modèle mental des réactions de l'interface.

5.1 Introduction

Nous percevons le monde extérieur par nos sens, vision, toucher, audition, perception d'accélération, et à un degré moindre, odorat. Ces sens sont des interfaces entre le monde et notre corps. Ils reçoivent et interprètent des "projections" du monde qui ont une structure beaucoup plus simple que le monde réel. Il est donc possible de concevoir des techniques qui produisent une illusion du monde extérieur grâce à la synthèse de ces projections. Ces projections synthétiques sont calculées par ordinateur, et sont présentées aux sens par l'intermédiaire de systèmes d'interface attachées au corps (lunettes de visualisation, écouteurs, matrices d'éléments vibrants). Pour cette raison, Jaron Lanier, qui a créé l'expression "réalité virtuelle", appelle ces systèmes d'interface "vêtements informatisés" (computerized clothing) [34]. Ces projections synthétiques n'ont pas besoin d'atteindre un réalisme parfait, car l'opérateur peut s'adapter au nouveau *langage* qui lui est présenté. Mais le réalisme est désirable pour augmenter le confort et l'efficacité de l'interaction, et diminuer la période d'adaptation.

Les projections du monde changent en fonction de notre pose dans le monde. Lorsque nous déplaçons la tête, de nouvelles parties de la scène deviennent visibles. De plus, nos mains se déplacent devant nous, et peuvent modifier la scène, soit en déplaçant les objets, soit en saisissant un outil pour changer la couleur, la forme d'un objet. La réalité virtuelle est donc nécessairement *interactive*, puisque le monde synthétique doit changer en réponse à certains déplacements du corps (c'est en fait cette caractéristique qui distingue ce domaine de l'informatique graphique pure). Des mécanismes doivent donc être en place pour continuellement détecter ces déplacements du corps et les transmettre à l'ordinateur, qui calcule les projections synthétiques en fonction de ces déplacements. En particulier, il est important de détecter la position de la main, pour permettre le déplacement d'objets synthétiques et l'utilisation d'outils virtuels, et la position de la tête, pour créer une synthèse visuelle réaliste dépendant du point de vue.

5.2 Réalité virtuelle et réalisme visuel

Une attention toute particulière doit être donnée à la synthèse de la projection visuelle du monde. Une grande partie de notre expérience de la réalité semble provenir de notre interprétation de la projection du monde sur la rétine de chaque œil. Une indication de l'importance de cette expérience est fournie par la proportion du cerveau dédiée à l'analyse et à l'interprétation de ces projections: environ 50%. Cette large proportion est sans doute aussi une indication que ce problème d'analyse n'est pas facile. En effet les photons qui entrent en collision avec la rétine après avoir rebondi sur un objet transmettent peu d'information concernant cet objet: la façon dont ils ont rebondi sur l'objet et la direction d'où ils viennent. Par temps clair, il est difficile de savoir si ces photons ont été réfléchi par un objet proche ou lointain. Le cerveau paraît résoudre ce problème par l'utilisation d'un grand nombre de modules, chaque module analysant des composantes différentes de l'information disponible, et corroborant les conclusions des autres modules. Le cerveau transmet à son propriétaire une impression de satisfaction qui est d'autant plus grande qu'un plus grand nombre de modules est d'accord sur l'interprétation des données d'entrée, que ces entrées proviennent du monde réel ou synthétique. Cette satisfaction est liée à l'impression de *réalisme* de l'expérience visuelle. Dans un ordre croissant de niveau de satisfaction visuelle liée au réalisme, on peut citer les expériences visuelles suivantes:

- La visualisation d'une projection perspective d'une scène 3D sur un écran d'ordinateur
- La visualisation d'une scène 3D sur un écran d'ordinateur avec des lunettes 3D à occlusion alternative des yeux par cristaux liquides
- La visualisation d'une scène 3D sur un écran d'ordinateur avec les lunettes 3D, avec changement de vue quand l'opérateur déplace la tête (obtenu par un dispositif de mesure de déplacement de la tête)
- La visualisation avec des écrans montés sur la tête (masque de visualisation) donnant à l'opérateur la possibilité d'explorer le monde virtuel par des rotations de la tête sur 360 degrés

Lorsqu'on pense aux mécanismes d'acquisition d'information 3D par le cerveau, le mécanisme qui vient d'abord à l'esprit est celui qui tire cette information des deux images de la vision binoculaire. Mais la distance entre les yeux est faible, si bien que cet effet est en fait surtout utile pour les parties de la scène qui sont à moins de 10 mètres. Il existe un mécanisme plus souple et plus puissant. Le cerveau peut comparer une image avec l'image obtenue après un déplacement latéral de la tête. Constamment et sans intention consciente, la tête se déplace latéralement pour obtenir l'information 3D. Ce mouvement fournit un effet de stéréovision active dans lequel la distance entre points de vue est choisie en fonction de la distance des points d'intérêt. Le déplacement latéral est amplifié lorsqu'une information plus précise de profondeur est nécessaire. On peut se surprendre en train de faire ces larges déplacements latéraux au volant pour évaluer les distances avant une manœuvre de parking entre deux voitures. En plus de l'information absolue fournie par l'effet stéréo de déplacement, une information immédiate entre les profondeurs relatives des objets de la scène est fournie par les déplacements apparents relatifs de ces objets. Dans un film sur écran géant, on peut remarquer que la caméra est très souvent en *traveling*, précisément pour fournir au spectateur un supplément de réalisme 3D en faisant appel à ce mécanisme de stéréo de déplacement.

Avec les lunettes 3D, l'opérateur a l'impression de voir flotter la scène devant l'écran d'ordinateur. S'il enlève les lunettes, il remarque que deux vues décalées sont affichées sur l'écran. Dans le dispositif le plus courant, ces vues sont en fait affichées en alternance et parviennent chacune à un seul œil grâce aux cristaux liquides contenus dans les verres des lunettes qui deviennent opaques en alternance à la même fréquence. Mais avec ce dispositif seul, l'opérateur ne peut voir aucune partie supplémentaire de la scène lorsqu'il déplace la tête latéralement; il acquiert donc l'impression que l'objet tourne comme pour refuser de laisser voir ses faces cachées. L'ordinateur ne peut recalculer des vues différentes de l'objet à présenter à l'opérateur à moins qu'on lui fournisse continuellement une mesure du déplacement de l'opérateur.

Un niveau de satisfaction visuelle très supérieur est fourni par un système qui recalcule les vues présentées à l'opérateur alors qu'il se déplace latéralement. La position des lunettes dans l'espace doit être recalculée constamment, et les vues de la scène corrigées

en conséquence. L'impression que l'objet tourne pour éviter de montrer ses parties cachées disparaît. Par contre s'il enlève les lunettes et les déplace latéralement devant l'écran avec la main, l'opérateur peut être surpris de voir la scène tourner en sens inverse. Mais avec les lunettes sur les yeux, ce n'est qu'avec cette correction qu'il interprète l'objet comme fixe. Il a alors l'impression extrêmement vive d'avoir une scène réelle fixe devant lui. Cette impression est étonnante et bien sûr difficile à décrire, et nous a convaincu que la synthétisation de cet effet de parallaxe est aussi essentielle que la vision binoculaire dans la construction d'un effet 3D réaliste.

Enfin, avec un masque comportant des écrans de visualisation montés sur la tête, l'opérateur a l'impression supplémentaire d'être complètement entouré par un monde virtuel [52]. En effet, l'opérateur est alors libre de tourner la tête dans toutes les directions, et les images présentées devant ses yeux sont recalculées en fonction de ces déplacements. Ce système doit aussi utiliser un traqueur pour la mesure continue de la pose de la tête dans l'espace.

5.3 Ecran fixe ou masque de visualisation?

Comme on l'a indiqué, on peut obtenir une impression 3D très vive pour des objets représentés sur un écran fixe, grâce à des lunettes 3D et un dispositif de calcul continu de la pose de la tête. Les lunettes sont presque aussi légères que des lunettes de soleil, et ne coupent pas de la réalité "réelle" des collègues et du téléphone car elles sont plus transparentes que des lunettes de soleil et n'ont pas besoin de supprimer la vision latérale et la lumière venant des côtés. L'opérateur a devant les yeux un écran de résolution 1024×1024 qui couvre environ 45 degrés de son champ de vue. Le champ de vue humain est d'environ 120 degrés, avec environ 90 degrés de vision binoculaire [20]. L'écran couvre donc environ la moitié de la partie binoculaire du champ. Par contre, avec un masque de visualisation, l'immersion dans le monde virtuel est totale, et le monde virtuel qui peut être appréhendé est plus vaste. Les modèles les plus récents comportent une paire d'écrans couleur à cristaux liquides de résolution 640×240, qui sont vus par l'intermédiaire de lentilles grand-angle montées à environ 5 mm des écrans. Ces lentilles introduisent une déformation "en oreiller" des images. Pour compenser cette déformation, une déformation inverse "en tonneau" est créée

sur les écrans. Ces lentilles sont grand-angle de façon à couvrir la totalité du champ de vue de l'opérateur pour augmenter l'effet d'immersion. Des jupes souples de type "masque de plongée" empêchent la lumière extérieure de venir interférer avec la vision des écrans.

Ces masques de visualisation ne pourront remplacer un système à écran fixe que lorsqu'ils deviendront aussi légers et aussi faciles à mettre en place que des lunettes, et lorsqu'ils offriront une résolution au moins deux fois plus fine, tout cela pour un prix raisonnable (moins de 5000 F). Les techniques actuelles ne permettent pas d'atteindre ces objectifs. En conséquence, les domaines d'application des deux systèmes sont actuellement différents. Dans le monde réel, il y a de nombreuses tâches dans lesquelles l'opérateur n'a pas besoin de regarder autour de lui et derrière lui. Dans ces tâches, l'utilisateur fait face à une surface ou un volume de travail fixes: tour du potier, hotte du chimiste, établi du modéliste, machine-outil, etc.... Du fait de leur résolution beaucoup plus fine, les systèmes utilisant un écran fixe sont mieux adaptés aux tâches dans lesquelles l'opérateur interagit avec des objets dans un volume de travail fixe. On peut appeler ce type de réalité virtuelle "réalité virtuelle de bureau" (Desktop VR). Notons aussi qu'un pilote regarde surtout dans la direction de sa trajectoire, c'est-à-dire devant lui à travers le pare-brise. Pour voler à travers un espace virtuel, un système à écran fixe fournit une meilleure résolution dans la direction du vol qu'un masque de visualisation.

Par contre le masque de visualisation donne une sensation d'immersion supérieure, en partie parce que le monde virtuel paraît présent tout autour, et aussi par une impression de mouvement relatif plus réaliste, car la vision périphérique qui est très sensible au mouvement reçoit le flux d'images approprié. On peut appeler ce type de réalité virtuelle "réalité virtuelle immersive".

Dans les sections qui suivent, nous présentons des exemples d'applications spécifiques pour chacune des deux familles de réalité virtuelle sous forme de scénarios.

5.4 Applications de la réalité virtuelle de bureau

- Un ingénieur en hydraulique travaille sur une nouvelle pompe. Il porte des lunettes à cristaux liquides qui comportent un dispositif de détection de pose, et tient dans la

main un pointeur 3D. La pompe est représentée en 3D dans un programme qui modélise les mouvements des pièces mécaniques et simule l'écoulement du fluide. Il augmente la vitesse de la pompe et note une diminution soudaine du rendement. Il rend le corps de la pompe invisible, pour révéler un écheveau de particules se déplaçant le long de lignes de courant, et changeant de couleur en fonction de la pression. Il déplace la tête sur la gauche et note une région dans laquelle les couleurs pulsent rapidement du rouge au bleu. Déplaçant la tête sur la droite, il note une deuxième zone de pulsation. Il note que les deux zones de pulsations ne sont pas en phase. La pompe est dans un mode de résonance défavorable à cette vitesse. L'ingénieur fait réapparaître les parois de la pompe, et ne visualise que la partie de l'écoulement entre les deux régions pulsantes; avec le pointeur 3D, il prend un outil de modélisation pour élargir le passage entre les deux régions, tout en observant la différence de phase entre les deux régions. Il fait ainsi disparaître le phénomène de résonance [44].

- Une analyste en finances surveille les actions de 600 companies étrangères en temps réel. Chaque action est représentée par une tige, qui oscille plus rapidement d'avant en arrière quand les prix montent, et de gauche à droite quand les prix descendent. Avec toutes ces tiges, les données ressemblent à un champ de blé qui oscille doucement dans la brise. Soudain, elle remarque des oscillations latérales plus rapides des épis plantés dans la région du Guatemala. Elle survole cette région et coupe ces épis avec son pointeur 3D, un geste qui transmet immédiatement l'ordre de vendre les actions de cette région. Dix minutes plus tard, CNN annonce un coup d'état militaire au Guatemala et c'est la panique sur les actions de ce pays; mais ses clients n'ont souffert que des pertes limitées parce qu'elle a pu détecter le problème avant qu'il soit reporté [44, 59].

Ce dernier exemple illustre la puissance des outils de visualisation appliqués à des données abstraites. Du fait qu'une bonne partie du cerveau est impliquée de près ou de loin dans l'interprétation des impressions visuelles, la mise en images de données abstraites et non physiques est un *codage* qui permet d'utiliser le canal de transmission le plus efficace, ayant la bande passante la plus large entre les données elles-même et les zones cérébrales capables de raisonner sur ces données [59]. La densité d'information qui peut être détectée sur un

écran peut être augmentée de façon remarquable par l'utilisation de perspectives 3D et d'animations interactives [48], de sorte que des relations entre données qui seraient très difficiles à détecter dans des tables de nombres deviennent visuellement évidentes.

5.5 Applications de la réalité virtuelle immersive

L'approche précédente transporte le monde virtuel dans l'espace de travail devant l'opérateur. Par contre dans l'approche de la réalité virtuelle immersive, l'opérateur est transporté au milieu du monde virtuel. Les scènes dans lesquelles nous sommes immergés dans le monde réel à cause de leur échelle (bâtiments, Grand Canyon) sont naturellement mieux simulées par un système de réalité virtuelle immersive que par un système de réalité virtuelle de bureau.

- Une architecte montre à son client un projet pour une nouvelle maison. Le client a vu des plans de la maison auparavant, mais avec le masque de visualisation, il trouve la répartition des volumes à son goût. Elle fait grandir les arbres pour lui montrer que même dans dix ans les feuilles ne tomberont pas dans la piscine. Le client va voir la piscine de plus près, et imagine qu'il vient de se baigner et veut rentrer dans la maison. Mais la porte la plus proche donne dans la salle de séjour. L'architecte reconnaît que c'est un problème. Elle déplace la piscine vers la chambre qui a une grande salle de bain, perce une porte extérieure pour la salle de bain et ajoute une courte allée entre la salle de bain et la piscine. Le client donne alors le feu vert pour la construction [44].

- Dans le roman "Snow Crash" [51], l'auteur, Neal Stephenson, imagine un monde virtuel partagé, le Métavers (un méta-univers), auquel les gens accèdent à partir de leur ordinateur par l'intermédiaire du réseau mondial de fibres optiques. Le système est donc un Internet ou Minitel de troisième génération. Ce monde se présente comme la Grande Rue d'une ville. Les gens y apparaissent sous une forme virtuelle appelée "avatar", qui se déplace en fonction de leurs mouvements. Ils peuvent bien sûr voir les autres dans leur masque de visualisation. Des développeurs achètent du terrain virtuel pour construire des bâtiments virtuels, des parcs d'attraction, des enseignes publicitaires.

Les propriétaires peuvent créer leur propres règles et lois physiques. De nombreux bâtiments sont dédiés à l'échange d'information entre personnes qui ont des intérêts communs. Les expressions faciales des participants sont détectées par leurs ordinateurs et représentées sur les avatars, ce qui facilite les négociations.

5.6 Applications en cours de développement

- Archéologie: Les objets et structures sont modélisés et placés en 3D dans la position où ils sont découverts. Il est alors plus facile de découvrir des tendances globales et de déduire les zones inexplorées qui méritent une excavation.

- Production de films: Les positions des acteurs, des éclairages, des caméras, sont optimisées en réalité virtuelle avant le tournage [67].

- Aéronautique: Les ingénieurs concevant un nouveau réacteur d'avion vérifient que toutes les pièces à inspecter ou changer régulièrement sont facilement accessibles [45].

- Chirurgie: Le chirurgien voit l'image 3D de son scalpel dans l'image 3D (tomographie, résonance magnétique) du cerveau du patient. Cette image peut parvenir à un seul œil, alors que l'autre œil voit la tête du patient(*réalité augmentée*) [60].

- Ligne de montage: Lorsque l'apprenti regarde un objet réel, une image générée par ordinateur se superpose à l'image réelle dans ses lunettes pour fournir des instructions spécifiques: positions des trous à percer sur la pièce, positions des composants à assembler sur la pièce, séquence des opérations (réalité augmentée) [45].

5.7 La main et l'outil

Comme on l'a mentionné plus haut, il est important de détecter la position de la main. Certains systèmes détectent aussi les flexions des doigts grâce à un gant [65] ou à un "exosquelette" articulé attaché aux doigts par des bandes Velcro [66]. L'avantage de ces détections est de permettre la représentation d'une main dans l'espace virtuel avec des doigts

dans une position qui reflète celle des doigts de l'opérateur. La métaphore de l'interface est alors très facile à comprendre et à apprendre pour l'opérateur: il peut déplacer la main vers un objet virtuel, et le saisir en fermant la main une fois que celle-ci intersecte avec un objet.

Toutefois, nous ne sommes pas convaincus pas que cet avantage justifie la complexité accrue du système. La détection des mouvements des doigts nécessite un nombre encombrant de détecteurs de déplacement angulaire (fibres optiques à transmission variable ou charnières à résistance électrique variable, etc...), de fils électriques, de données et de calibrations [44]. En outre, la représentation de la main est un des curseurs possibles les plus complexes pour symboliser sur l'écran la capacité à déplacer des objets. En fait, il a été amplement démontré pour les interfaces graphiques 2D que la pression d'un seul bouton suffit pour transmettre la mise en œuvre d'une *fonction*, et que la nature de la fonction elle-même peut être définie au préalable par le choix d'un outil dans une palette [49]. Nous pensons que ce paradigme s'applique aussi bien en 3D. Une fourchette, des pincettes ou des baguettes sont des outils familiers plus simples et plus spécifiques pour représenter la fonction de saisie des objets. D'autre part, la pression d'un bouton pour mettre en œuvre la fonction de saisie ne demande pas de compétence supérieure à la fermeture de la main. Comme en 2D, le déplacement d'objets est loin d'être la seule fonction utile dans le monde virtuel. Pour chaque fonction, il est préférable de représenter l'outil qui symbolise le plus spécifiquement possible la fonction. Par exemple, dans un programme de sculpture où l'opérateur peut sélectionner des gouges ou des truelles, il est plus utile de représenter la gouge que la main qui tient la gouge.

5.8 Métaphores pour l'interaction

La détection de la position instantanée de la tête de l'opérateur est généralement utilisée uniquement dans le calcul du point de vue (il serait préférable d'utiliser les positions des yeux eux-mêmes, mais la détection des positions des yeux est plus difficile que celle de la tête [56]). Le détecteur de position de la main – le pointeur 3D – est donc le canal principal par lequel l'opérateur peut modifier le monde virtuel. Dans le monde réel, nous nous sommes habitués à certaines relations de cause à effet entre les déplacements de la main et leurs effets. Par exemple, lorsque nous tenons un objet et plions le poignet, l'objet tourne avec la main.

Par contre, si nous sommes en moto et faisons le même mouvement du poignet, la moto accélère et le paysage défile plus vite. Dans le cas d'une interface d'ordinateur, il est utile que la *métaphore* présentée à l'opérateur soit claire, et explicitée par un curseur d'écran de forme spécifique. Si l'opérateur ne sait pas qu'il est dans une métaphore "moto", il sera sans doute surpris de voir le paysage changer quand il tourne le poignet. Une métaphore est fournie à l'opérateur pour lui permettre d'utiliser un modèle cognitif familier, et l'aider à comprendre et prédire les réactions du système à ses actions [58, 49] de façon à réduire la période d'adaptation.

Une métaphore que nous avons déjà discutée est la métaphore "outil". Lorsque le pointeur 3D est en mode " outil ", la scène des objets présentés sur l'écran est fixe lorsque le pointeur se déplace, et l'opérateur voit un curseur se déplacer dans la scène; lorsque le curseur entre en contact avec un objet, l'opérateur peut exprimer le désir (par exemple en pressant un bouton du pointeur) de laisser le curseur déplacer l'objet ou transformer l'objet d'une manière qui caractérise la fonction de l'outil.

Ce mode "outil" peut être utilisé de manière limitée pour naviguer, si par exemple l'opérateur peut accrocher un outil "ventouse" au sol ou au mur de la scène et tirer la scène vers lui. Il est sans doute possible d'explorer un labyrinthe de cette façon, mais si les distances sont grandes par rapport à l'amplitude de déplacement produite, ce mode d'opération est inefficace.

Une deuxième métaphore, "caméra dans la main", paraît à première vue utile pour l'exploration de la scène. Dans ce mode d'opération, la scène présentée à l'opérateur correspond à ce qu'il verrait si la vue était prise par une caméra tenue à la main. Un déplacement de la caméra produit un déplacement de la scène dans la direction opposée. Mais le champ d'exploration est sévèrement limité par la métaphore. Si l'opérateur veut voir la scène à partir de derrière, il devra étendre le bras pour placer la caméra au delà des objets les plus lointains, puis faire un mouvement de 180 degrés du poignet pour tourner la caméra vers lui. Cette contorsion peut produire des problèmes de détection avec les capteurs optiques et acoustiques à cause de l'occlusion du pointeur 3D par la main. De plus, dans cette position, un déplacement latéral de la caméra vers la gauche produit un déplacement peu intuitif de la scène vers la gauche par rapport à l'opérateur. Cette métaphore fut jugée déroutante et

d'utilité limitée par les utilisateurs [58].

Les expériences montrent que pour l'exploration d'une scène, une métaphore "véhicule volant" est beaucoup plus utile. Ware et Osborne [58] ont créé un mode d'opération dans lequel les translations de la main produisent des changements de *vitesse* linéaire du véhicule, et les rotations produisent des changements de vitesse angulaire. Le véhicule accélère lorsque le pointeur 3D est déplacé vers l'avant, et stoppe avant d'accélérer en marche arrière lorsque le pointeur est déplacé vers l'arrière. Les vitesses sont liées au cube du déplacement pour permettre de passer plus facilement des déplacements rapides aux accostages précis. Les auteurs mentionnent que ce mode d'opération est jugé efficace par les utilisateurs pour explorer des domaines complexes tels que des labyrinthes; toutefois les utilisateurs avec une expérience de pilote ont du mal à s'adapter à un avion qui recule.

5.9 En résumé

La réalité virtuelle peut constituer un outil puissant d'interaction avec des données complexes, qui code les données de façon visuelle et tridimensionnelle pour obtenir la bande passante la plus large avec les zones cérébrales capables de raisonner sur ces données. L'interaction avec les données est effectuée surtout par l'intermédiaire de déplacements de la main et de la tête, qui doivent être détectés et transmis à l'ordinateur par des traqueurs. Pour cette interaction, deux métaphores se révèlent particulièrement utiles et complémentaires: la métaphore "véhicule volant" pour explorer de grandes distances de la scène, jusqu'au domaine local intéressant; et la métaphore "outil", pour manipuler les objets dans ce domaine local.

Chapitre 6

Traqueurs pour la Réalité Virtuelle

Nous définissons quatre groupes de paramètres de performance permettant d'évaluer les traqueurs utilisés en réalité virtuelle; puis nous décrivons les différents principes physiques appliqués par les traqueurs, et nous comparons les contraintes imposées par ces principes. Nous concentrons enfin notre attention sur les traqueurs utilisant la capture et l'analyse d'images obtenues par une ou plusieurs caméras. Pour ces techniques nous définissons les compromis et les choix qui nous paraissent importants et qui ont guidé la conception du traqueur optique que nous avons construit.

6.1 Paramètres de performance des traqueurs

Comme on l'a vu au chapitre précédent, la création d'interfaces 3D interactives requiert l'utilisation d'un traqueur qui détecte la position dans l'espace soit de la main elle-même, soit d'un pointeur tenu dans la main. De plus, le réalisme visuel peut être clairement amélioré si un second traqueur détecte la position de la tête de l'opérateur, de façon que le changement de position de la tête produise un changement d'image correspondant au changement de point de vue de la scène.

Meyer et al. définissent une nomenclature complexe de paramètres de performance pour

ces traqueurs [40]. Nous avons simplifié cette nomenclature et conservé les paramètres suivants: précision absolue et résolution, temps de réponse, robustesse, et rayon d'action. Ces paramètres nous servent par la suite à comparer les différentes techniques, et aussi à définir les techniques de filtrage que nous pensons préférables pour notre projet.

6.1.1 Précision absolue et résolution

La précision absolue est l'amplitude de l'intervalle dans lequel la position correcte est garantie de se trouver. Si un traqueur a une précision en translation de 1 cm, la position actuelle est dans l'intervalle ± 1 cm autour de la position mesurée. La résolution est le *changement* de position minimal que le dispositif peut détecter. Ces deux quantités sont dépendantes; mais la distinction est utile, surtout pour les traqueurs qui mesurent uniquement des déplacements relatifs d'une pose à la suivante: un tel traqueur pourrait détecter des changements de position très fins, mais produire une précision absolue qui se détériore à mesure que les erreurs de mesure de déplacement relatif d'une position à la suivante s'accumulent.

Les erreurs de mesure ont une composante aléatoire dans l'intervalle de résolution, et si cet intervalle est grand, elles peuvent produire des images qui vacillent de manière désagréable à moins qu'un filtrage soit effectué. Mais un filtrage diminue la précision et la résolution du traqueur et augmente son temps de réponse.

6.1.2 Temps de réponse

Le temps de réponse total est la somme de trois intervalles:

1. Le temps d'acquisition et de calcul du traqueur
2. Le temps de transmission des données du traqueur à l'ordinateur
3. Le temps de calcul et d'affichage de l'ordinateur

Le problème est aussi compliqué par l'utilisation d'un filtrage des mesures dans le traqueur ou l'ordinateur. Ce filtrage peut être nécessaire pour éviter que les erreurs de mesure produisent des images vacillantes. Mais ce filtrage introduit une inertie de réponse qui est perçue

comme un temps de réponse supplémentaire. Pour prendre un exemple extrême, un traqueur peut avoir un temps de réponse théorique de 10 ms dans sa prise en compte de la plus récente mesure, mais se comporter avec l'inertie d'un traqueur qui a un temps de réponse au moins cinq fois plus grand, si le filtrage consiste à moyenner les dix dernières mesures. L'adaptation à un traqueur devient difficile si le temps de réponse dépasse 60 ms [40].

Nous pensons qu'une quantité importante dans cette analyse est le rapport signal de déplacement/ bruit de vacillement de l'image, qui doit rester élevé. En d'autres termes, le vacillement peut être déplaisant quand le pointeur ou la tête sont presque immobiles, mais il ne se remarque pas dans les mouvements rapides. D'autre part, il est important d'avoir le minimum d'inertie dans les mouvements rapides. On peut donc obtenir un compromis utile avec peu de vacillement dans les mouvements lents et peu d'inertie dans les mouvements rapides, par un filtrage qui est appliqué uniquement pour les déplacements lents et qui est supprimé dès que les déplacements deviennent rapides.

Le traqueur optique que nous avons construit a un temps de réponse total de 30 à 45 ms:15 ms pour le traqueur lui-même en l'absence de filtrage, 5 ms pour la transmission à l'ordinateur, environ 10 ms pour le calcul de la pose et de la nouvelle image, et de 0 à 15 ms jusqu'à l'instant de l'affichage sur l'écran.

6.1.3 Robustesse

La robustesse est une mesure de la fiabilité du traqueur dans l'environnement considéré. Par exemple, un traqueur qui se sert d'une caméra couleur pour distinguer les couleurs de petits disques sur un pointeur 3D n'est sans doute pas robuste dans un environnement où les utilisateurs ont des cravates à pois multicolores, mais devient plus robuste dans un laboratoire où les gens ne portent pas de cravate. Certains traqueurs, qui semblent à première vue peu robustes parce qu'ils sont sensibles aux occlusions, peuvent être optimisés pour que les occlusions se produisent surtout en dehors du domaine utile de fonctionnement. Par exemple, pour un traqueur optique détectant la position de la tête d'un opérateur devant un écran, il est facile de positionner une caméra de façon qu'une occlusion se produise uniquement lorsque l'opérateur tourne la tête à un angle qui ne lui permet plus de voir

l'écran, auquel cas l'occlusion n'a pas d'importance.

6.1.4 Rayon d'action

La robustesse et la précision diminuent généralement quand la distance entre les capteurs et les émetteurs augmente. Ce sont donc ces facteurs qui limitent le rayon d'action. Dans les applications qui nécessitent un grand rayon d'action, la solution est de multiplier les émetteurs ou les capteurs, mais le logiciel devient compliqué. Un système utilisant un gyroscope pourrait en principe avoir un très grand rayon d'action, mais les mesures de pose nécessitent des intégrations qui se détériorent avec le temps; des mesures absolues sont donc nécessaires de temps en temps pour recalibrer le gyroscope par une autre technique utilisant des capteurs et des émetteurs, et on est alors ramené au problème précédent.

6.2 Types de traqueurs de position

6.2.1 Capteurs mécaniques

Le mode de détection de position qui paraît le plus simple est une détection mécanique, obtenue en connectant la main ou la tête à des points fixes de référence par l'intermédiaire de câbles avec des enrouleurs, ou de barres articulées. Des encodeurs angulaires transmettent en continu les angles des enrouleurs ou des articulations à un algorithme qui utilise la géométrie de l'assemblage pour calculer la matrice de pose de la main ou de la tête. Sutherland, un des pionniers de la réalité virtuelle, construisit en 1968 un système de barres entre le plafond et son masque de visualisation. La confiance n'était pas absolue, certains appelaient ce système " l'Epée de Damoclès" [40, 52].

Les traqueurs mécaniques peuvent avoir une bonne précision et un temps de réponse court avec des techniques de mesure faciles à maîtriser telles que des codeurs angulaires optiques. Les connexions mécaniques ont tendance à limiter leur rayon d'action et à interférer avec certains mouvements. Mais ce sont sans doute les seuls systèmes capables de faire ressentir une force en réponse à certains événements. Par exemple, un pointeur mécanique 3D muni de servo-moteurs dans ses articulations peut transmettre à l'opérateur des forces

proportionnelles à l'attraction électrique entre une molécule fixe et la molécule qu'il manipule, lui permettant de découvrir de nouvelles combinaisons chimiques [7].

6.2.2 Capteurs magnétiques

Le traqueur le plus populaire en réalité virtuelle (Polhemus [33, 4]) comprend une antenne fixe émettrice qui crée un champ tournant dans l'espace au moyen de trois solénoïdes dont les axes sont mutuellement perpendiculaires. Le capteur est aussi constitué de trois solénoïdes perpendiculaires. L'orientation du capteur est déduite des différences de phases entre le champ émis et les courants induits, et la translation est déduite de l'intensité de ces courants. La précision est maximale à 80 cm, environ 1% en translation et 1 degré en rotation. Le temps de réponse est d'environ 10 ms avec une fréquence de mesure de 120 Hz dans les nouveaux modèles, qui échantillonnent les signaux induits et utilisent des techniques digitales. Les modèles commercialisés jusqu'à présent avaient une réponse généralement jugéee trop lente, autour de 100 ms. Un problème majeur avec ce type de traqueur est que les objets métalliques environnants (console de visualisation, meubles de bureau) deviennent des antennes secondaires qui perturbent le champ. Les traqueurs doivent être calibrés pour compenser ces distorsions du champ. D'autres types de traqueurs magnétiques tentent d'éviter ces perturbations en utilisant des impulsions électromagnétiques rectangulaires et en ne mesurant l'intensité du champ induit que lorsque l'état stationnaire est atteint (Ascension Technology [40]). Mais il faut alors faire des corrections pour le champ magnétique terrestre. Dans tous les cas, le traitement du signal est complexe, et cette complexité se traduit par un prix relativement élevé (plus de 15000 F). Mais les traqueurs magnétiques présentent l'avantage d'être insensibles aux occlusions non métalliques entre émetteurs et capteurs.

6.2.3 Traqueurs acoustiques

Les traqueurs acoustiques utilisent des ultrasons (au–dessus de 20 kHz). Le principe est de mesurer les distances entre des émetteurs et des microphones. Avec trois émetteurs fixes et un microphone monté sur l'objet mobile (pointeur 3D, lunettes, masque de visualisation), on peut déterminer la position du microphone dans l'espace par triangulation. Avec trois

microphones montés sur l'objet mobile, on peut déterminer la position et l'orientation de l'objet.

Certains traqueurs émettent des *impulsions* ultrasoniques et calculent les distances par le temps du trajet de ces impulsions aux microphones. Chacun des émetteurs émet une impulsion à tour de rôle, et les trois microphones détectent les fronts de ces impulsions. La fréquence de mesure est limitée par le fait que, pour éviter les confusions, il faut attendre quelques millisecondes que les échos produits par la salle diminuent avant d'émettre une nouvelle impulsion.

La deuxième méthode possible consiste à émettre des signaux ultrasoniques continus, et à mesurer les différences de phase dans les signaux reçus par les microphones. Chaque émetteur produit une fréquence spécifique. Ces mesures de distance sont ambiguës, puisqu'à une distance décalée d'une longueur d'onde on observe la même différence de phase. A 40 kHz, une longueur d'onde est environ 8 mm. Le traqueur doit donc faire des mesures relatives, chosissant parmi les distances possibles celles qui sont les plus proches de celles de la mesure précédente. Les premières mesures doivent donc être prises à une position de calibration connue exactement, ce qui n'est pas très pratique. Cette méthode a l'avantage de fournir des mesures de pose à plus haute fréquence que la méthode utilisant des impulsions. Les deux méthodes sont sensibles aux occlusions entre émetteurs et microphones, mais avec la deuxième méthode, l'absence de mesures due à une occlusion produit des mesures absolues fausses par la suite. Ce n'est pas nécessairement un problème pour un pointeur 3D; mais pour la position de la tête, il est important d'obtenir des mesures absolues, surtout dans le domaine de la réalité augmentée, qui superpose dans le champ de vision de l'opérateur le monde virtuel et le monde réel. C'est sans doute la raison pour laquelle les efforts actuels de développement semblent porter surtout sur la première méthode (pour une description de ces développements, voir [40]).

6.2.4 Traqueurs optiques

Les traqueurs optiques ont reçu une attention plus grande que les autres systèmes. Une des raisons est que les caméras vidéo sont des capteurs sophistiqués et rapides, qui ont un

prix modeste du fait de leur statut de produit de grande consommation. Par contraste, les techniques décrites précédemment doivent faire appel à des capteurs spéciaux. De plus, les caméras noir et blanc sont généralement très sensibles à la lumière infrarouge, si bien que des éléments émetteurs infrarouges sont faciles à détecter. Les traqueurs optiques partagent certains inconvénients avec les capteurs acoustiques: ils ne fonctionnent qu'en l'absence d'occlusions entre caméra et objet à traquer, et la précision diminue lorsque la distance entre caméra et objet augmente.

Certaines recherches ont porté sur la détection optique de la position de la main nue de l'opérateur [38, 68], et la reconnaissance de gestes. Cette approche séduisante est à l'heure actuelle coûteuse, du fait que les algorithmes sont complexes et ne peuvent tourner assez rapidement que sur des ordinateurs puissants.

Les objectifs du projet Mandala sont plus modestes [49]: une caméra pointée vers l'opérateur détecte sa silhouette et superpose son image avec celle de la scène sur l'écran. L'opérateur peut interagir avec les objets de l'image en déplaçant sa silhouette. Par exemple, des instruments de percussion sont représentés sur l'écran, et lorsque la silhouette d'une main atteint une cymbale sur l'écran, le système produit le son de la cymbale [49]. La silhouette de l'opérateur est détectée facilement grâce à un arrière–plan bleu devant lequel l'opérateur doit se trouver. Les interactions se font par déplacement latéral de la silhouette; l'interaction avec une scène tridimensionnelle nécessiterait le calcul des paramètres de pose du corps de l'opérateur, ce qui est beaucoup plus difficile.

Dans les véhicules bruyants et comportant de nombreuses pièces métalliques, les traqueurs optiques sont plus robustes que les traqueurs magnétiques ou acoustiques. En particulier, cette technologie est utilisée pour détecter la position des casques de pilotes dans les avions de chasse de façon à communiquer des informations graphiques dans leur champ de vision et à leur permettre de viser sans lâcher les commandes. Des brevets d'invention d'origine américaine et israélienne témoignent de cette activité [17, 64]. Le brevet de Egli [17] est particulièrement intéressant. Il semble avoir été mis en application dans un système de cockpit développé par Honeywell. Quatre sources LED (Light Emitting Diodes) sont montées sur le casque du pilote et sont éclairées séquentiellement. Les lignes joignant une des sources aux trois autres sont mutuellement perpendiculaires. La pose de cet arrangement est calculée

à partir des images des sources prises par une seule caméra. Cette géométrie est une de celles que nous avons utilisées pour nos pointeurs 3D, et la découverte tardive de ce brevet nous a forcés à éliminer cet arrangement de la description d'un de nos brevets [13]. Mais la méthode de calcul présentée dans le brevet d'Egli ne peut s'appliquer qu'à cet arrangement, tandis que les méthodes de calcul de pose présentées dans nos brevets [13, 14, 15] et dans les chapitres précédents s'appliquent à de nombreuses configurations de sources et exigent généralement moins d'étapes de calcul.

Plusieurs traqueurs en vente actuellement [60, 40] utilisent deux ou trois caméras et appliquent des algorithmes de stéréométrie pour obtenir la position des points caractéristiques d'un objet dans l'espace. Ces points sont généralement des sources de lumière infrarouge. Ces traqueurs n'utilisent la connaissance de la géométrie relative de ces points que dans l'étape finale de calcul de la pose de l'objet. Les algorithmes pour cette approche sont beaucoup plus compliqués que les algorithmes que nous proposons pour une seule caméra. Un traqueur utilisant plusieurs caméras est en principe capable de produire une pose plus précise qu'une méthode utilisant une seule caméra. Mais cet avantage peut être également obtenu en combinant les poses calculées à partir de chaque caméra par l'algorithme POSIT, sans faire appel aux méthodes de triangulation généralement appliquées aux traqueurs à caméras multiples.

Des chercheurs de l'Université de North Carolina ont développé un traqueur optique qui permet à un opérateur de se déplacer dans une salle [57]. Dans ce type d'application, il est moins coûteux d'avoir un grand nombre de sources de lumière fixées au plafond de la salle, et des caméras sur le casque de l'opérateur, plutôt qu'un grand nombre de caméras fixes et des sources sur le casque. Les sources sur le plafond sont différenciées par une activation à tour de rôle. La pose du casque est calculée en utilisant la méthode de Church, qui est une méthode itérative utilisée en photogrammétrie [61] pour calculer la pose d'une caméra à partir de triangles de points. Cette méthode est à notre avis d'utilisation délicate car elle fournit une seule pose alors que les triangles ont toujours deux (et parfois quatre) poses possibles pour une image donnée [19, 11]. Mais l'idée de placer les caméras sur le casque est spécialement prometteuse avec le développement récent de caméras de masse très faible intégrées sur une seule puce [69].

6.3 Critères de conception pour un traqueur optique

En théorie, de nombreux algorithmes de vision artificielle capables de déterminer la pose d'un objet à partir d'une image ou d'une séquence d'images pourraient être appliqués pour trouver la pose d'un casque ou d'un pointeur 3D. On pourrait même utiliser un scanner à laser pour obtenir une image de distance de la scène, détecter l'image de l'objet dans la scène, et trouver la position de l'objet à partir de la connaissance de sa forme [43]. Mais le problème qu'on essaie de résoudre ici est différent du problème que les chercheurs en vision artificielle essaient de résoudre. Pour ces chercheurs, le problème est généralement: Comprendre la scène, c'est-à-dire reconnaître certains objets. Par contre ici, le problème peut être énoncé de la manière suivante:

> Optimiser à la fois l'objet et l'algorithme de vision pour que la pose de l'objet soit fournie de manière robuste et précise dans le temps minimal avec le minimum de ressources matérielles et informatiques.

Certains choix pour atteindre ce but sont clairs. D'abord, pour minimiser le temps de calcul, il faut minimiser l'analyse de bas niveau de l'image. Par exemple si certaines parties de l'objet sont très lumineuses, le reste de la scène peut être éliminé par seuillage. De petites taches claires sur l'objet demandent moins de calcul que des lignes caractéristiques claires, et un calcul de pose utilisant des lignes [16, 18] serait sans doute plus compliqué. Ces taches claires peuvent être obtenues par de petites sources de lumière infrarouge, ou par des éléments convexes réfléchissant la lumière d'une source infrarouge unique.

De plus, une seule caméra est suffisante pour calculer la pose d'un objet si le calcul utilise l'information concernant l'arrangement géométrique des sources de lumière. Ce choix réduit le coût, d'abord en éliminant une des caméras, ensuite en éliminant la structure qui maintient les deux caméras dans une position relative fixe. Pour un pointeur manuel 3D, le calcul de pose n'a pas besoin d'être très précis, parce que les résultats d'un déplacement sont corrigés par une rétroaction utilisant l'observation des résultats. Par contre, pour un traqueur de position de la tête, une précision élevée est plus importante, surtout pour éviter les oscillations de la visualisation dues aux erreurs de pose sans avoir recours à des filtrages

qui introduisent des délais. Une meilleure précision est obtenue si on augmente la distance entre les sources lumineuses. Mais l'arrangement des sources devient alors encombrant, plus encombrant en fait qu'une caméra miniature [69]. Il semble donc préférable de monter la caméra sur la tête et de monter l'arrangement des sources en position fixe au–dessus de la console de visualisation.

D'autres choix sont plus difficiles. Par exemple, la méthode de mise en correspondance la plus simple consiste à éclairer les sources de lumière de l'objet l'une après l'autre de manière que chaque image contienne une seule tache lumineuse, et comme on l'a vu c'est la méthode qui a été choisie dans plusieurs systèmes existants. Toutefois, avec une caméra courante produisant 60 images par seconde, la détection de quatre sources exige quatre images, c'est-à-dire 66 millisecondes. Il faut ajouter à cela les temps de transmission vers l'ordinateur, de calcul et d'affichage des images, ce qui conduit à un temps de réponse de 80 ms ou plus. Comme on l'a vu, ce temps de réponse est très perceptible et peu désirable. Pour diminuer le temps de réponse, on doit utiliser une caméra spéciale plus rapide et plus coûteuse que les caméras courantes. Pour cette raison nous préférons la méthode qui consiste à laisser toutes les sources lumineuses continuellement éclairées, et à résoudre la correspondance entre les taches lumineuses par le logiciel. On peut alors obtenir des poses toutes les 17 ms, pour un temps de réponse total d'environ 45 ms, avec une caméra de série [13, 14, 15].

L'utilisation d'une caméra de série ouvre la possibilité de créer un produit grand-public bon marché. En effet, le marché des caméras s'étend très rapidement, et les prix des caméras sont en baisse rapide. Certaines entreprises proposent déjà des caméras de haute qualité intégrées sur une seule puce pour moins de 400 F [69]. Il faut aussi noter que des ordinateurs grand-public tels que le Macintosh seront bientôt vendus équipés de caméras.

6.4 En résumé

Nous avons passé en revue les techniques qui sont appliquées dans les traqueurs utilisés en réalité virtuelle, et nous avons défini des paramètres de performance pour ces traqueurs. Par comparaison aux autres systèmes, il est relativement facile d'obtenir des mesures à fréquence assez élevée et temps de réponse faible avec des caméras vidéo courantes. Ces

caméras sont en train de devenir bon marché. Le système que nous décrivons dans la suite utilise une seule caméra et profite de ces avantages pour fournir une solution peu coûteuse. Mais il y a des risques d'occlusion et de superposition des sources de lumière; les effets de ces événements sur le comportement du curseur doivent être minimisés; nous expliquons au chapitre 8 les méthodes que nous avons appliquées pour minimiser ces problèmes. Comme pour les traqueurs magnétiques et acoustiques, la résolution et la précision diminuent quand la distance entre la caméra et les sources de lumière augmente (voir chapitres 2 et 3).

Chapitre 7

Architectures pour un Traqueur Optique d'Interface 3D

Dans ce chapitre, nous décrivons les composants électroniques nécessaires pour la construction d'un traqueur optique permettant l'interaction avec une scène d'objets tridimensionnels représentés par un ordinateur. Nous décrivons le circuit que nous avons construit, ainsi que d'autres circuits conçus par la suite. Ce chapitre résume certaines parties d'une demande de brevet américain [15] par l'auteur en collaboration avec Yukio Fujii, Image and Media System Laboratory, Hitachi, Ltd.

7.1 Vue d'ensemble et principes du système

La figure 7.1 présente une vue d'ensemble d'un système dans lequel l'opérateur utilise un pointeur 3D optique pour interagir avec des objets représentés sur une console d'ordinateur. Le pointeur comporte quelques sources de lumière. Une caméra est positionnée à côté de l'écran d'ordinateur et fait face à l'opérateur. Cette caméra est équipée d'un filtre et produit un signal vidéo qui est transmis à une unité de détection. Cette unité analyse les images codées dans le signal vidéo et calcule les coordonnées des centres des taches claires créées

par les sources de lumière du pointeur 3D. Ce sont ces coordonnées qui sont transmises à l'ordinateur, par un câble série en format RS232, avec une fréquence de 60 Hz, correspondant à la fréquence de transmission d'images par le signal vidéo de la caméra (section 7.4). L'ordinateur calcule la position et l'orientation ("pose") du pointeur 3D à cette fréquence à partir de ces coordonnées. Utilisant ces poses calculées pour le pointeur 3D, l'ordinateur rafraîchit constamment une image perspective d'un curseur virtuel 3D, et affiche cette image sur l'écran. Nous appelons cette image *curseur d'écran*. Nous examinons à présent chacun des éléments du système plus en détail.

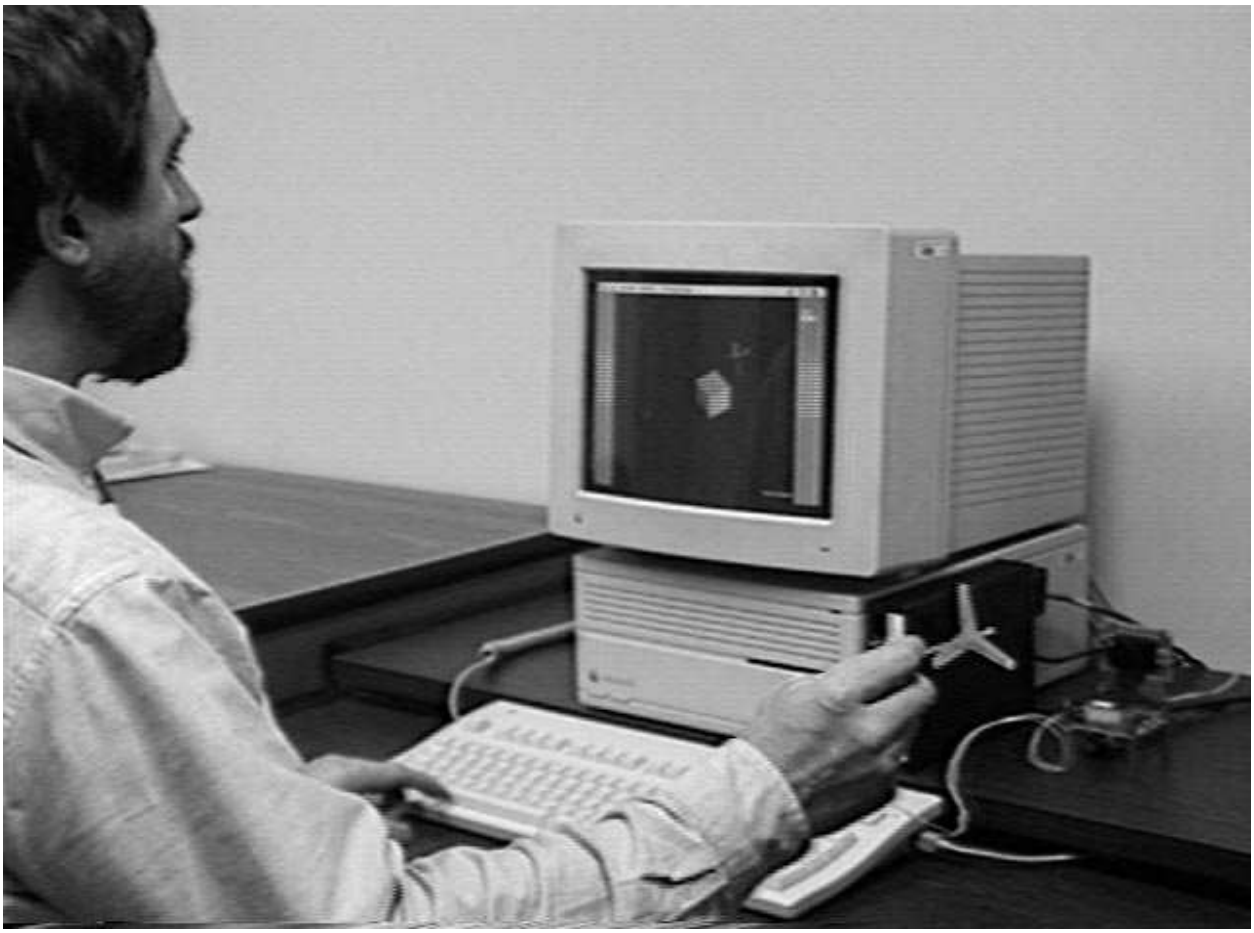

Figure 7.1: Pointeur 3D utilisant la vision artificielle. La boîte noire à droite de l'ordinateur détecte les images des sources de lumière et transmet leurs coordonnées à l'ordinateur.

7.2 Pointeur 3D

Le pointeur 3D (figure 7.2) comprend un corps que l'opérateur peut tenir dans sa main, et une armature comportant quelques sources de lumière. L'armature est de préférence fine et transparente pour réduire les chances d'occlusion entre les sources et la caméra. Les sources sont par exemple des ampoules à incandescence miniatures d'environ 2 mm de diamètre, qui comme toutes les ampoules à incandescence émettent une bonne partie de leurs radiations dans le domaine infrarouge. On peut aussi utiliser des diodes lumineuses (Light Emitting Diodes, ou LED) émettant dans le domaine infrarouge, mais l'énergie infrarouge produite par les diodes courantes est plus faible que celle produite par des lampes miniatures.

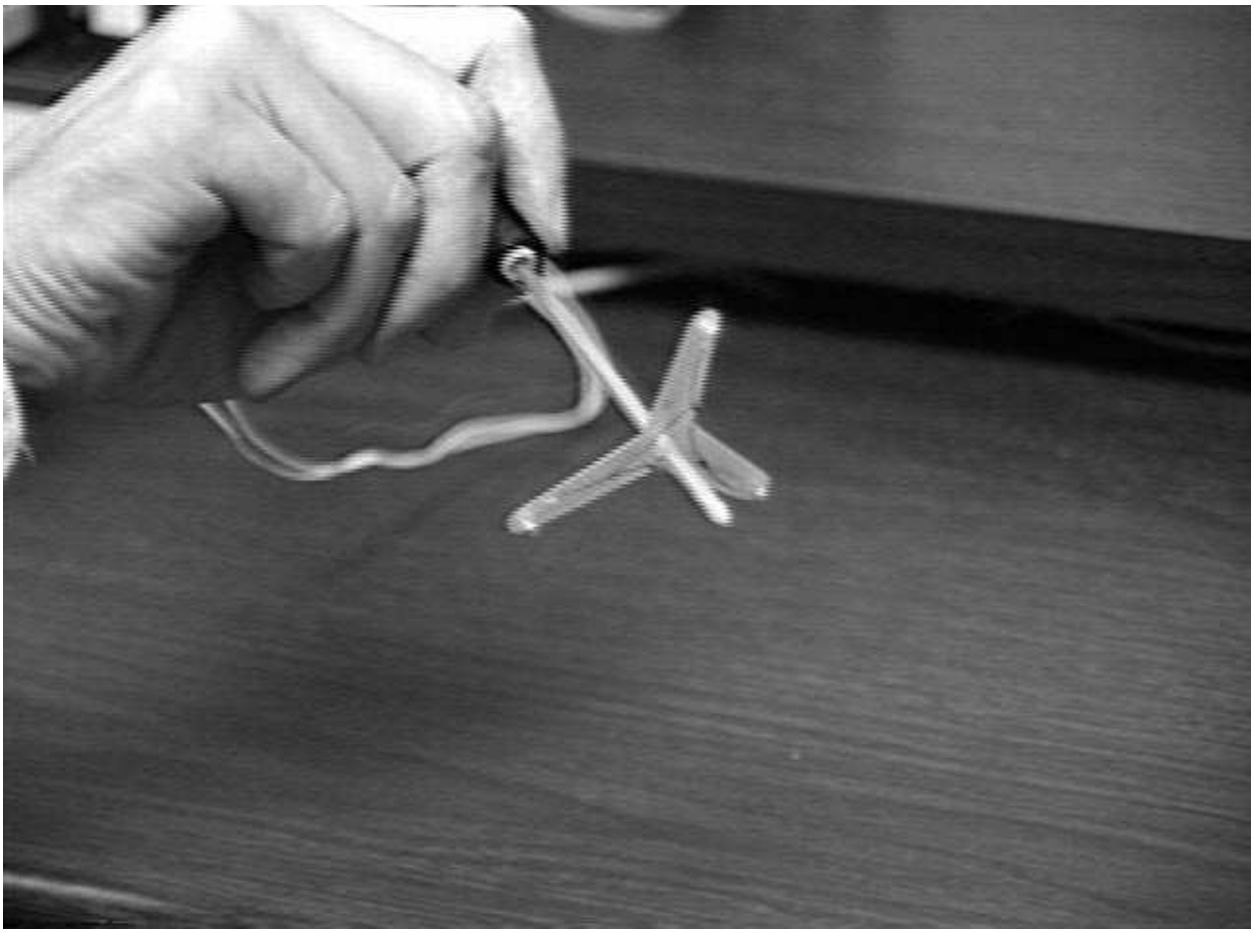

Figure 7.2: Pointeur 3D utilisant quatre ampoules incandescentes miniatures.

La figure 7.3 illustre une autre construction possible pour un pointeur 3D: les sources de lumière sont les extrémités de guides de lumière transparents, par exemple de fibres

optiques en plastique de diamètre 1 mm environ. Ces fibres transmettent la lumière émise par une source principale située à l'intérieur du corps du pointeur. La source principale est une ampoule à incandescence, une diode lumineuse ou une diode laser. Cette source est alimentée par des piles par l'intermédiaire d'un bouton interrupteur comme sur la figure 7.3, ou par un câble d'alimentation comme sur la figure 7.1.

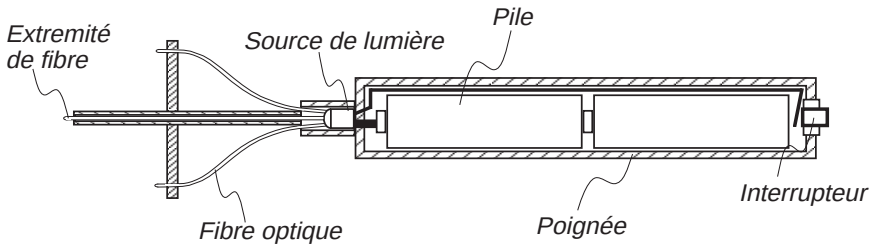

Figure 7.3: Pointeur 3D utilisant des fibres optiques pour créer des sources de lumière secondaires à partir d'une source primaire.

Il est souhaitable que les sources de lumière soient clairement visibles à partir d'angles de vue divers. Toutefois, l'extrémité d'un guide de lumière semble émettre la plus grande partie de l'énergie lumineuse dans la direction de son axe si la fibre se termine par une face circulaire plane obtenue en coupant simplement la fibre à angle droit par rapport à son axe. On peut donner à cette extrémité une forme conique (comme la mine d'un crayon bien taillé) avec du papier de verre fin; cette forme améliore considérablement l'émission de lumière dans les directions latérales par rapport à l'axe de la fibre.

Par comparaison aux sources de lumière créées par des ampoules ou des diodes, les fibres optiques permettent d'obtenir des points lumineux très fins. Les avantages comprennent la réduction des chances de voir les images de ces sources se superposer, et une précision améliorée de position détectée. Par contre, si ces images ne couvrent qu'un ou deux pixels à courte distance de la caméra, il y a un risque accru que ces sources ne soient plus visibles à grande distance.

7.3 Caméra

Le capteur CCD (Charge-Coupled Device) de la caméra (figure 7.4) est choisi pour être plus sensible aux radiations infrarouges qu'à la lumière visible, de sorte que la réponse aux sources

de lumière du pointeur est prépondérante par rapport à la réponse à la lumière ambiante. Ainsi les taches claires créées dans l'image vidéo de la caméra sont beaucoup plus intenses que l'image de la scène ambiante. Cette nécessité n'exige pas l'utilisation de matériel exotique, car en fait la plupart des capteurs CCD de caméras noir-et-blanc semblent avoir leur pic de sensibilité dans le domaine infrarouge.

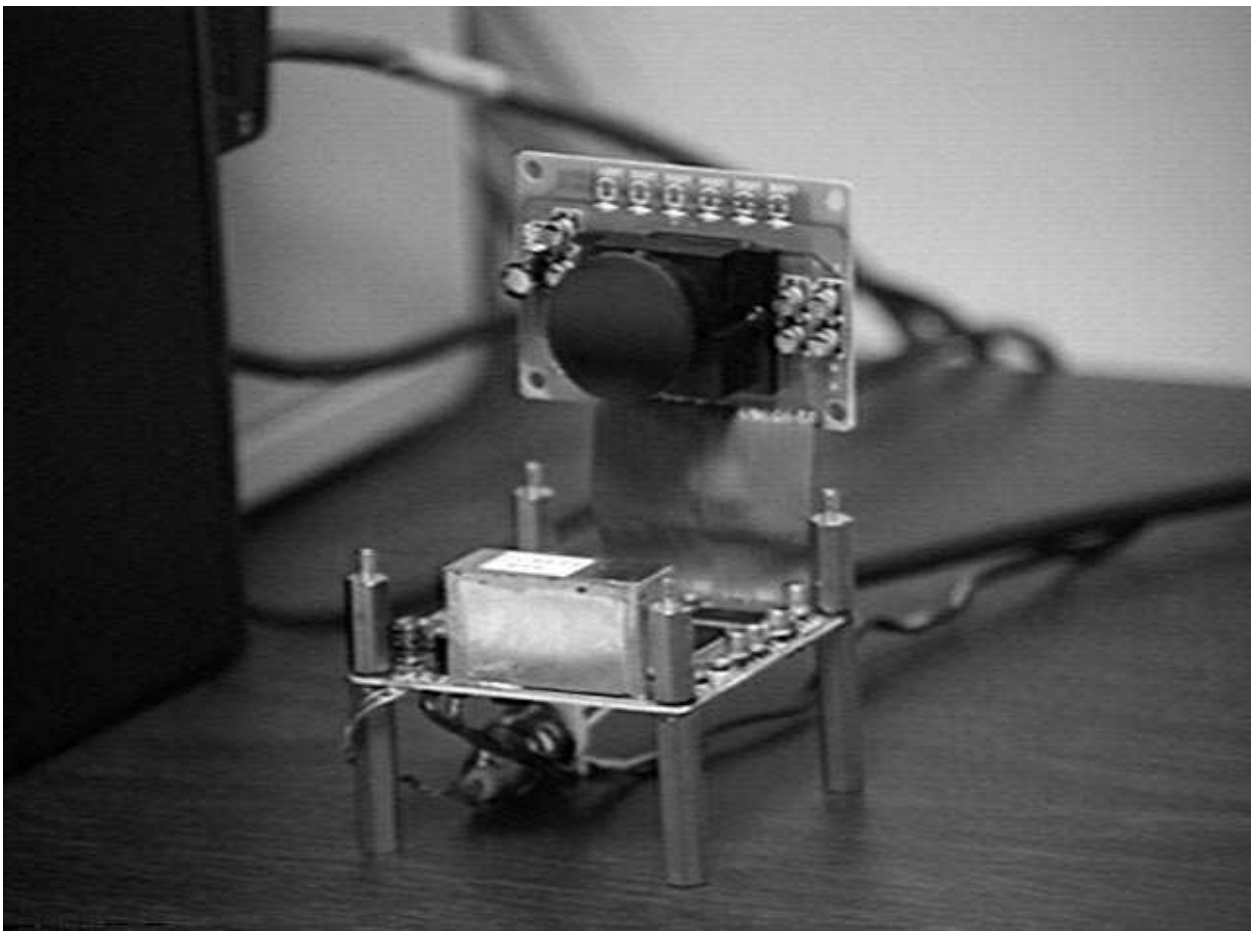

Figure 7.4: Nous avons surtout utilisé une caméra miniature Sanyo LC9931-EB05, ici sans son boîtier. La puce CCD comporte 486 lignes comprenant chacune 378 éléments sensibles. Le filtre noir monté devant l'objectif bloque la totalité du spectre visible (0.4 à 0.7 μm), et laisse passer les longueurs d'onde supérieures à 1 μm.

De plus, un filtre optique passe-bande est monté devant la lentille de la caméra. Ce filtre transmet sélectivement les longueurs d'onde émises par les sources de lumière et bloque les longueurs d'ondes de la lumière visible. La combinaison d'une caméra sensible à la lumière infrarouge et d'un filtre passe-bande rend la détection des taches claires et le rejet de la scène très faciles, pourvu que le soleil, une lampe à incandescence, ou un objet réfléchissant ces

sources de lumière chaude, ne soient pas dans le champ de la caméra. La figure 7.5 montre l'image vue par la caméra à travers le filtre.

Figure 7.5: Image produite en lumière ambiante par la caméra Sanyo équipée du filtre, en présence du pointeur 3D à quatre ampoules.

7.4 Signal vidéo

Il est nécessaire de se rappeler certaines caractéristiques du signal vidéo produit par une caméra vidéo noir-et-blanc pour comprendre les principes appliqués dans le système. La partie qui décrit les intensités lumineuses de l'image dans un signal vidéo est un signal *analogique* typiquement compris entre 0.4 V et 1 V. Ce signal analogique représente l'intensité à donner au faisceau d'électrons d'un moniteur de télévision lorsque ce faisceau balaie le phosphore de l'écran, ligne par ligne et de haut en bas. Une image vidéo complète est

généralement transmise en $1/25^e$ s (SECAM) ou $1/30^e$ s (U.S. National Television System Committee, ou NTSC). Une image complète est en fait transmise en *deux trames*, chaque trame en $1/50^e$ s ou 1/60 s, qui sont balayées l'une après l'autre de façon entrelacée sur l'écran. La figure 7.6 est une représentation schématique du signal vidéo correspondant à une trame d'image qui contient une seule tache claire. Cette figure montre qu'une portion de signal appelée période de retour de balayage horizontal est ajoutée par la caméra entre deux signaux de lignes de balayage. Cette période dure environ 10 μs, et contient une impulsion de synchronisation horizontale, pour laquelle le potentiel du signal tombe à un niveau proche de zéro, pour indiquer au moniteur de télévision qu'une ligne est terminée. Cette portion de signal laisse aussi le temps au faisceau de balayer l'écran en sens inverse pour se repositionner au début de la ligne suivante. De plus, à la fin de chaque trame, une portion de signal appelée période de retour de balayage vertical est ajoutée par la caméra pour laisser au moniteur de télévision le temps de se resynchroniser avec le signal vidéo. Cette période dure environ 1200 μs. Durant cette période, le faisceau d'électrons se replace en haut et à gauche de l'écran, prêt pour le balayage de la trame suivante.

7.5 Unité de détection des centres des taches claires

7.5.1 Contraintes d'opération

Dans la conception d'un système bon marché de détection des taches claires en temps réel, nous devons tenir compte des contraintes suivantes:

1. Pour détecter de façon précise les taches claires créées par les sources du pointeur dans l'image vidéo, il est préférable d'obtenir au moins 256 pixels sur chaque ligne vidéo par digitalisation. Chaque ligne vidéo est transmise par la caméra en 50 μs environ pour un signal vidéo NTSC, donc un nouveau pixel doit être digitalisé toutes les 200 νs (nanosecondes).

2. Les microcontrôleurs les plus courants et les moins coûteux sont des microcontrôleurs 8 bits, et les versions rapides ont une fréquence de 33 MHz. Ce type de microcontrôleur

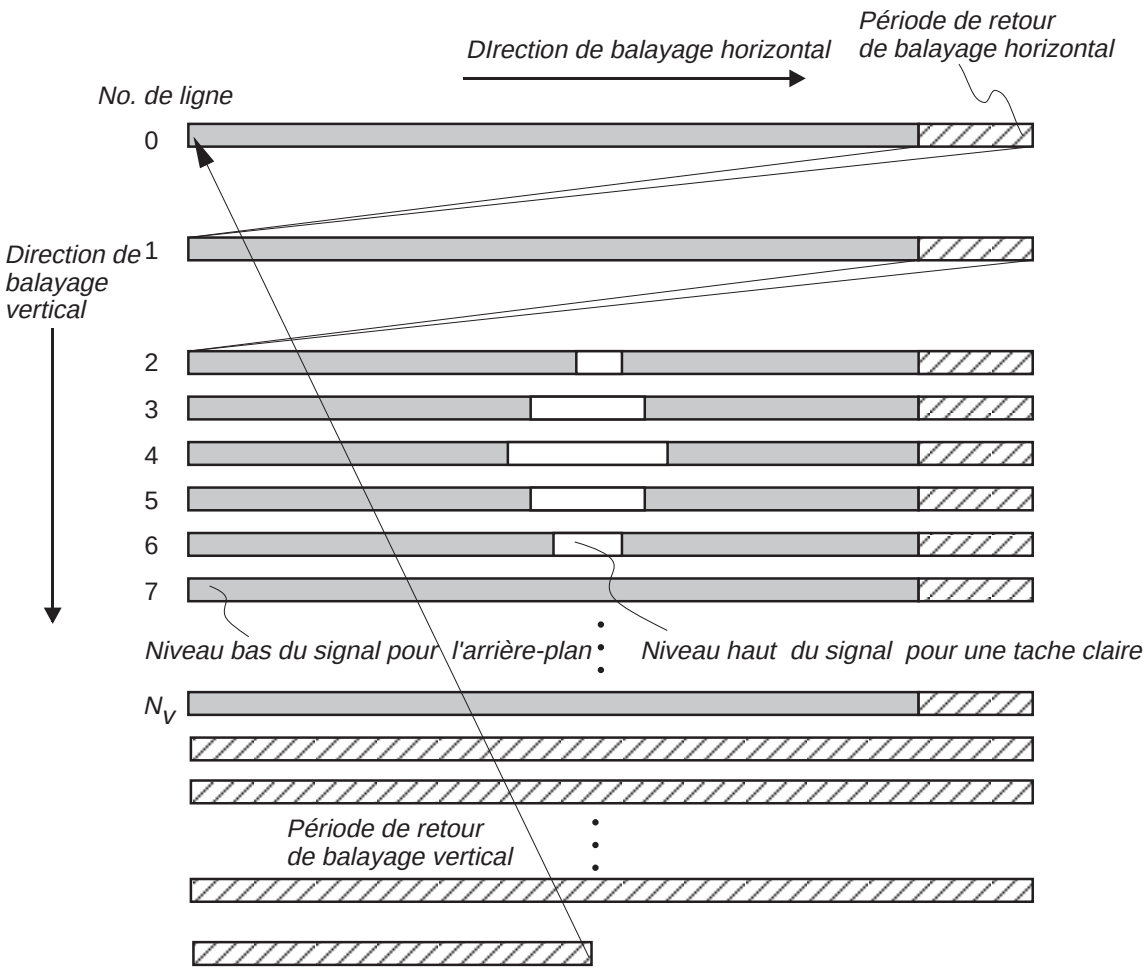

Figure 7.6: Représentation schématique du signal vidéo produit par la caméra.

exécute une instruction en 300 ou 600 νs. Par conséquent le microcontrôleur n'a pas le temps de transférer les pixels intéressants dans sa mémoire interne pendant que ces pixels sont produits, et encore moins d'opérer des opérations de regroupement entre ces pixels.

3. Par contre, nous avons vu qu'il y a des périodes de temps mort durant la transmission d'image: une période de retour de balayage horizontal d'environ 10 μs à la fin de chaque ligne, et une période de retour de balayage vertical de 1200 μs entre deux trames d'image.

4. Mais on ne peut pas simplement espérer stocker les 64K pixels d'une image en mémoire dans l'espoir de les lire et de les traiter durant les 1200 μs du retour de balayage vertical.

En effet, en temps de lecture seule, le processeur passerait environ 20 fois plus de temps, simplement pour lire les pixels.

Pour résoudre le problème de détection des taches claires en temps réel en dépit des difficultés décrites ci-dessus, il faut d'une part stocker les pixels en mémoire, au moment où ils sont produits et sans la possibilité d'utiliser le microcontrôleur (contrainte 2), d'autre part essayer de lire uniquement les pixels intéressants parmi tous les pixels stockés en mémoire (contrainte 4). Cette opération de lecture peut être effectuée soit durant la période à la fin de chaque image, soit si possible durant la période à la fin de chaque ligne. On doit aussi effectuer pendant cette période le calcul des centres des taches. Ce calcul consiste principalement à vérifier le chevauchement des limites à gauche et à droite des segments de taches dans deux lignes successives, et, en cas de chevauchement, à écrire que le centre de gravité de la nouvelle agglomération de segments est une combinaison linéaire, (1) du centre de gravité pour l'agglomération auquel appartient le segment de la ligne précédente et (2) du centre de gravité du nouveau segment. Deux architectures sont proposées; dans la première, on lit les pixels durant la période de balayage de retour vertical, et on saute les lignes de pixels qui ont été marquées vides au moment de la mise en mémoire; dans la deuxième, on utilise une mémoire FIFO (First In, First Out) pour enregistrer et lire uniquement les positions des rebords des taches claires dans chaque ligne, et on traite ces rebords durant l'intervalle de temps entre deux lignes. Nous considérons ces deux architectures dans les sections qui suivent.

7.5.2 Architecture avec stockage d'image

La figure 7.7 illustre une configuration de l'unité de détection des centres de taches claires qui met toute l'image en mémoire, avant de la traiter durant la période de retour de balayage vertical. Sur ce diagramme, le signal vidéo en provenance de la caméra 20 est transmis à un convertisseur A/D (Analog–Digital) et à un détecteur d'impulsion de synchronisation. Le convertisseur digitalise le signal vidéo, et la forme digitale est transmise au détecteur de niveau des taches (Spot Level Detector) 105. La fréquence d'échantillonnage du convertisseur est déterminée par le signal généré par l'horloge 103. Cette fréquence définit la résolution

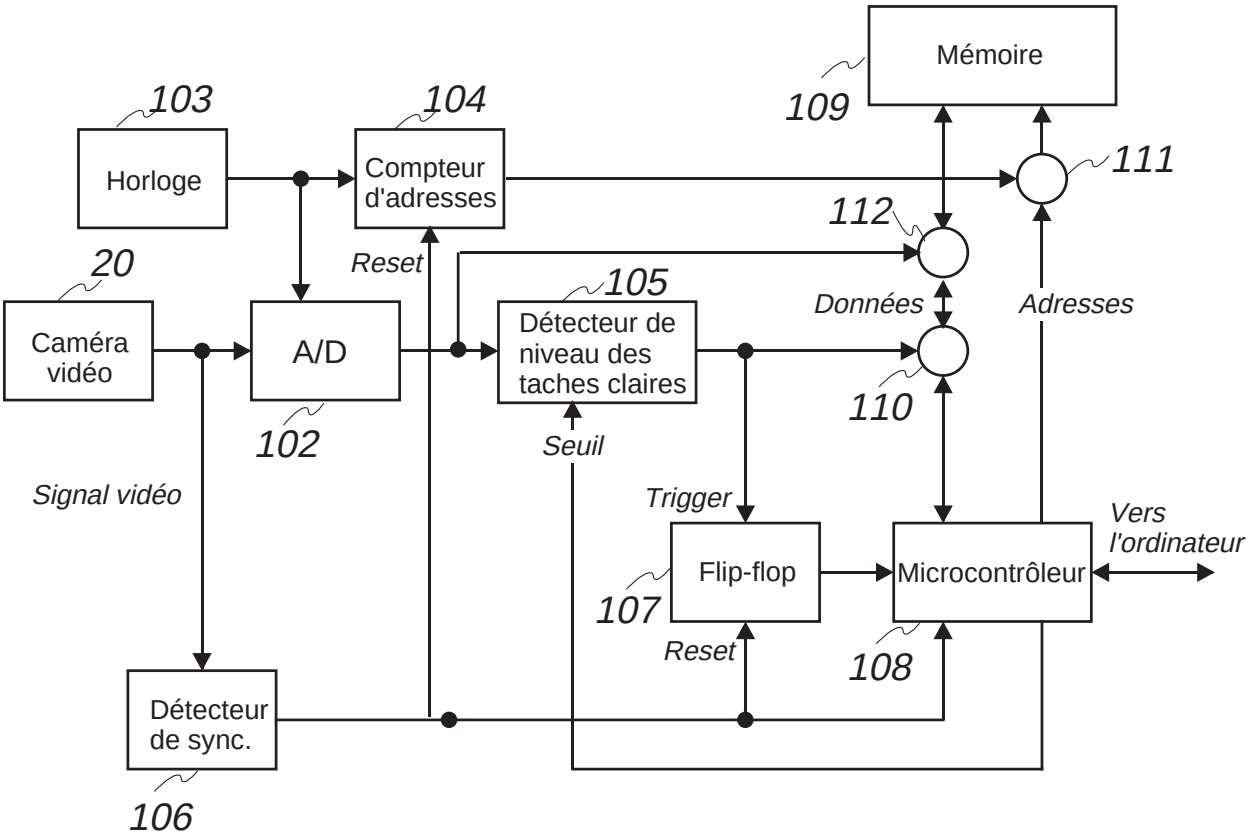

Figure 7.7: Unité de détection des centres de taches claires dans laquelle les lignes d'image vides sont marquées au moment de l'écriture en mémoire et sautées à la lecture.

horizontale de l'image digitale, c'est-à-dire le nombre de pixels (N_h) sur chaque ligne. Ce nombre de pixels doit être de préférence supérieur ou égal au nombre de lignes dans l'image (cette résolution verticale, N_v, est égale à 242 lignes par trame dans un signal NTSC) pour garantir un calcul précis de la position du pointeur 3D. Le signal d'horloge contrôle aussi le compteur d'adresses 104, qui génère une partie du nombre utilisé comme adresse pour stocker les valeurs des pixels. La sortie H_{sync} du détecteur d'impulsion de synchronisation 106 remet ce compteur à zéro au début de chaque ligne. Le détecteur de niveau des taches 105 distingue un pixel de tache claire par comparaison avec une valeur de seuil. La sortie de ce détecteur, appelée *niveau de pixel* dans la suite, est au niveau logique haut quand la valeur du pixel est supérieure à la valeur du seuil, et au niveau logique bas dans le cas contraire. La mémoire vive 109 (RAM) stocke un niveau de pixel passant par le sélecteur du circuit des données dans la cellule pointée par l'adresse provenant du compteur d'adresses 104; cette adresse est transmise à travers le sélecteur du circuit des adresses 111. Le circuit des données et le circuit

des adresses de la mémoire vive sont connectés au microcontrôleur 108 par l'intermédiaire du sélecteur du circuit des données 110 et du sélecteur du circuit des adresses 111. Le microcontrôleur 108 a ses séquences d'instructions et ses données stockées dans une mémoire morte (ROM), soit interne soit externe, et opère en réponse à ces instructions. Dans ce système, le microcontrôleur 108 peut soit aller lire les niveaux de pixels de la mémoire vive 109, soit écrire certains résultats dans la mémoire vive, en commutant à la fois le sélecteur du circuit des données 110 et le sélecteur du circuit des adresses 111, et en générant l'adresse de la cellule de mémoire de destination. Par l'utilisation des données de mémoire vive, de mémoire morte, et des signaux des ports d'entrée et de sortie, le microcontrôleur contrôle les opérations de mémoire, calcule les coordonnées des centres des taches claires et transmet les résultats à l'ordinateur principal par l'intermédiaire du câble série. Dans l'ordinateur principal, ces coordonnées sont combinées avec une matrice précalculée qui caractérise la géométrie des sources de lumière sur le pointeur 3D pour calculer la position et l'orientation de ce pointeur.

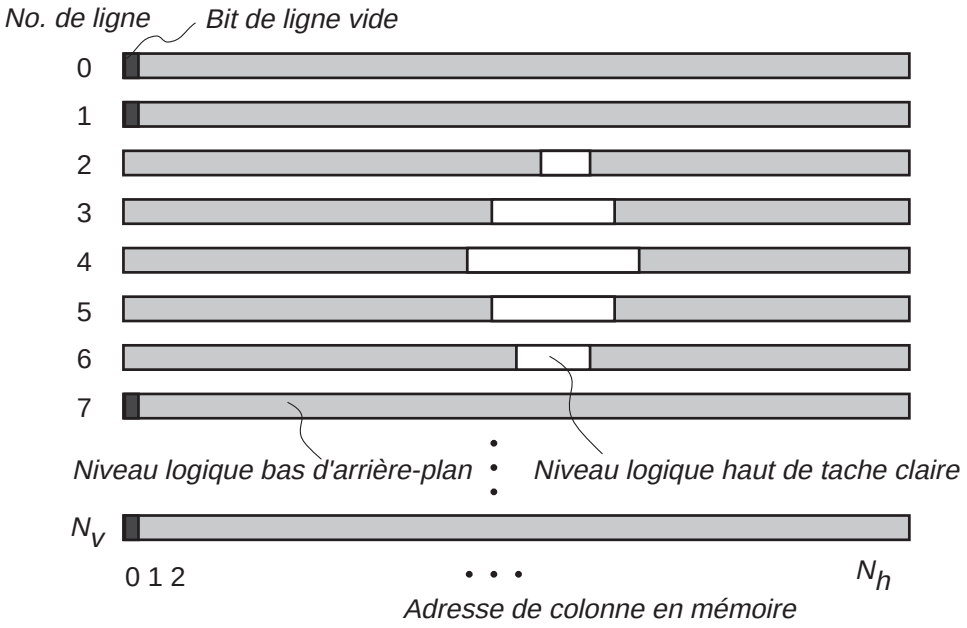

Figure 7.8: Distribution des données d'image dans la mémoire vive, comprenant une donnée enregistrée au début des lignes vides.

Comme on l'a mentionné, le microcontrôleur n'a pas le temps de lire tous les pixels stockés en mémoire. Dans le calcul des positions des taches claires, seules les lignes contenant des

pixels clairs exigent une lecture, et lors de la lecture de la mémoire vive, les lignes vides peuvent être négligées. Par conséquent, on doit donner au microcontrôleur la possibilité de lire un bit de signalisation lui indiquant s'il doit sauter la lecture d'une ligne ou non. Ce bit doit être créé au moment où la ligne est écrite en mémoire. Mais à cause de la haute fréquence d'échantillonnage, il est impossible de demander à un microcontrôleur tournant à 33 MHz de créer ce bit de signalisation, car il n'est pas assez rapide pour lire chaque niveau de pixel au moment où ce niveau est produit. Dans le système de la figure 7.7, le microcontrôleur laisse un flip-flop accomplir la tâche de détecter si une ligne lue est entièrement vide; si c'est le cas, le microcontrôleur constate que le flip-flop est resté dans son état initial, et marque un bit de signalisation dans la première cellule de la mémoire de cette ligne. La figure 7.8 montre le contenu de la mémoire vive 109 une fois que toutes les lignes vides ont été marquées avec un bit de signalisation 120. Sur cette figure, chaque bande horizontale de la figure correspond à une ligne de cellules de mémoire, et les zones blanches et grises correspondent respectivement à des niveaux de pixels hauts et bas stockés dans ces cellules. Sur la figure 7.7, le flip-flop 107 génère un niveau logique haut pour une ligne qui ne contient aucun pixel de niveau haut, et un niveau logique bas dans le cas contraire. Le flip-flop reçoit son entrée en provenance du détecteur de niveau de tache 105, et est réinitialisé par H_{sync} en provenance du détecteur d'impulsion de synchronisation 106. Le niveau de sortie du flip-flop est connecté à un port d'entrée du microcontrôleur.

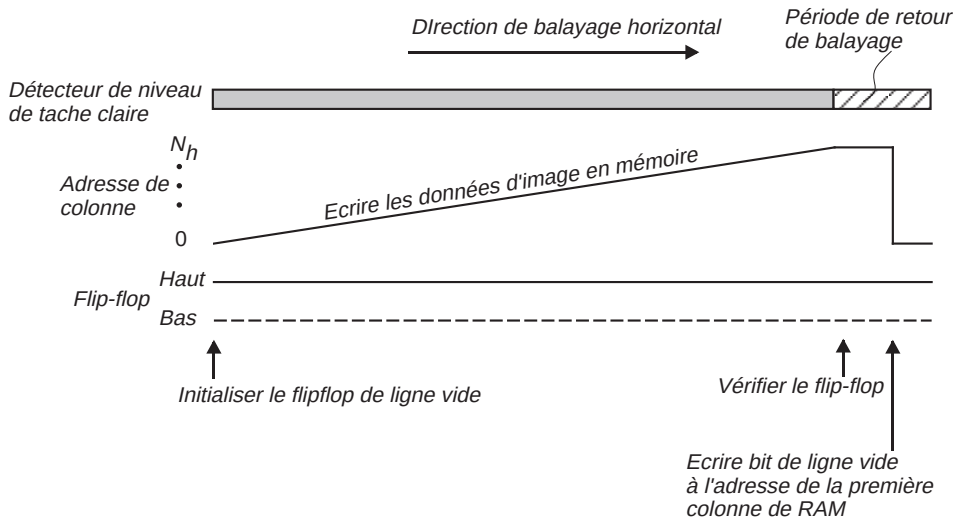

Figure 7.9: Détail de l'opération de marquage d'une ligne vide.

La figure 7.9 décrit l'opération du système quand la ligne est vide. L'axe horizontal représente la progression du temps de gauche à droite, et la barre en haut de la figure représente la sortie du détecteur de niveau de tache 105 pour la ligne. Alors que le temps s'écoule, l'adresse de colonne, qui est la sortie du compteur d'adresses 104, est incrémentée de 0 à N_h, et le niveau de pixel est écrit dans la mémoire vive 109 correspondant à l'adresse. Dès que la ligne est complétée, le microcontrôleur 108 vérifie la sortie du flip-flop 107. Ce flip-flop est au niveau haut au début de la ligne et reste au niveau haut puisqu'aucun pixel de niveau haut ne bascule le flip-flop dans ce cas. Puis le microcontrôleur 108 pointe à l'adresse 0 et réécrit un bit de signalisation de ligne vide à la place du niveau de pixel de la première colonne, durant la période de balayage de retour, qui dure environ 10 μs dans le standard NTSC; puis la ligne suivante est traitée. Durant la période de lecture de la mémoire, grâce au bit de signalisation de ligne vide, le microcontrôleur est capable de savoir si la ligne est vide simplement en lisant le premier site de mémoire de chaque ligne, au lieu d'avoir à lire toute la ligne.

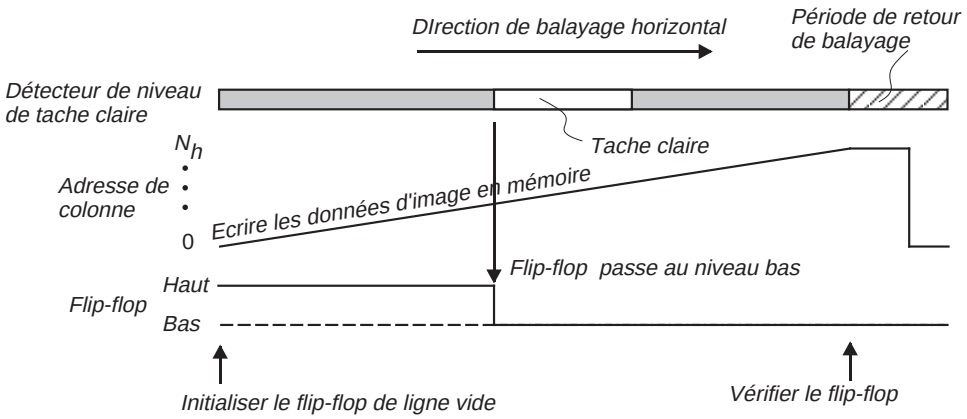

Figure 7.10: Détail de l'opération de détection de ligne vide lorsque des pixels clairs sont détectés dans une ligne.

La figure 7.10 montre le comportement du système quand la ligne contient un ou plusieurs pixels de niveau haut, représentés par les segments en blanc sur le dessin. Le flip-flop a été initialisé au niveau haut en début de ligne et l'adresse de colonne est incrémentée; l'apparition d'un niveau haut de pixel bascule le flip-flop 107 au niveau bas. Le microcontrôleur 108 vérifie le flip-flop et n'écrit pas de bit de signalisation dans ce cas.

Pour résumer, toutes les valeurs seuillées de pixels sont stockées dans la mémoire vive,

mais les lignes vides sont marquées comme illustré sur la figure 7.8, si bien que le microcontrôleur peut sauter la lecture des lignes vides dans la phase de calcul des centres des taches. Cette configuration permet au système de rafraîchir le curseur sur l'écran d'ordinateur à la fréquence de transmission des trames d'image dans le signal vidéo, c'est-à-dire répondre en temps réel aux déplacements du pointeur 3D.

Cette architecture présente l'avantage de pouvoir être aussi utilisée comme simple digitaliseur d'images. Cette fonction est obtenue par un sélecteur de circuit de données supplémentaire 112 et une déviation entre le convertisseur A/D 102 et le sélecteur 112 qui évite le détecteur de niveau de pixel 105. L'image brute peut être ainsi stockée en mémoire sans être seuillée. Avec la possibilité d'afficher les images vidéo brutes sur l'écran, l'opérateur peut faire le diagnostic de problèmes de traitement d'image, tels qu'un choix de niveau de seuil inapproprié ou une mauvaise focalisation de la caméra, et peut faire les calibrations nécessaires pour un fonctionnement optimal du système.

Pour notre construction, nous sommes partis d'un digitaliseur d'images décrit dans une série d'articles de Byte [60] et de Circuit Cellar Ink [41], et disponible en kit. Les changements nécessaires comprennent l'addition de composants spécifiques tels que le détecteur de niveau des pixels, le flip-flop, et certains sélecteurs de circuits, la mise en place d'un microcontrôleur et d'une horloge plus rapides, et l'installation d'un logiciel de contrôle et de détection des taches dans la mémoire morte utilisée par le microcontrôleur.

7.5.3 Architecture avec mémoire FIFO

Le circuit décrit ci-dessus est celui que nous utilisons. Il n'est pas vraiment optimal, car il met en mémoire les lignes vides qui ne sont pas utiles. Nous décrivons maintenant un circuit plus simple. Le seul désavantage apparent de ce circuit est qu'il ne peut pas fonctionner en capteur d'images brutes. Comme on le voit sur la figure 7.11, une mémoire FIFO (First In First Out) remplace la mémoire vive 109 de la figure 7.7. Ce type de mémoire ne comporte pas d'entrée d'adresse, mais comporte une entrée d'initialisation, et des entrées pour activer l'écriture et la lecture des données. Comme son nom l'indique, la mémoire FIFO enregistre les données dans leur ordre d'entrée lorsque l'entrée d'activation d'écriture est au niveau

haut. De plus, à chaque opération de lecture ou d'écriture, la mémoire FIFO incrémente son compteur d'adresses interne. Ce compteur est remis à zéro par l'entrée d'initialisation.

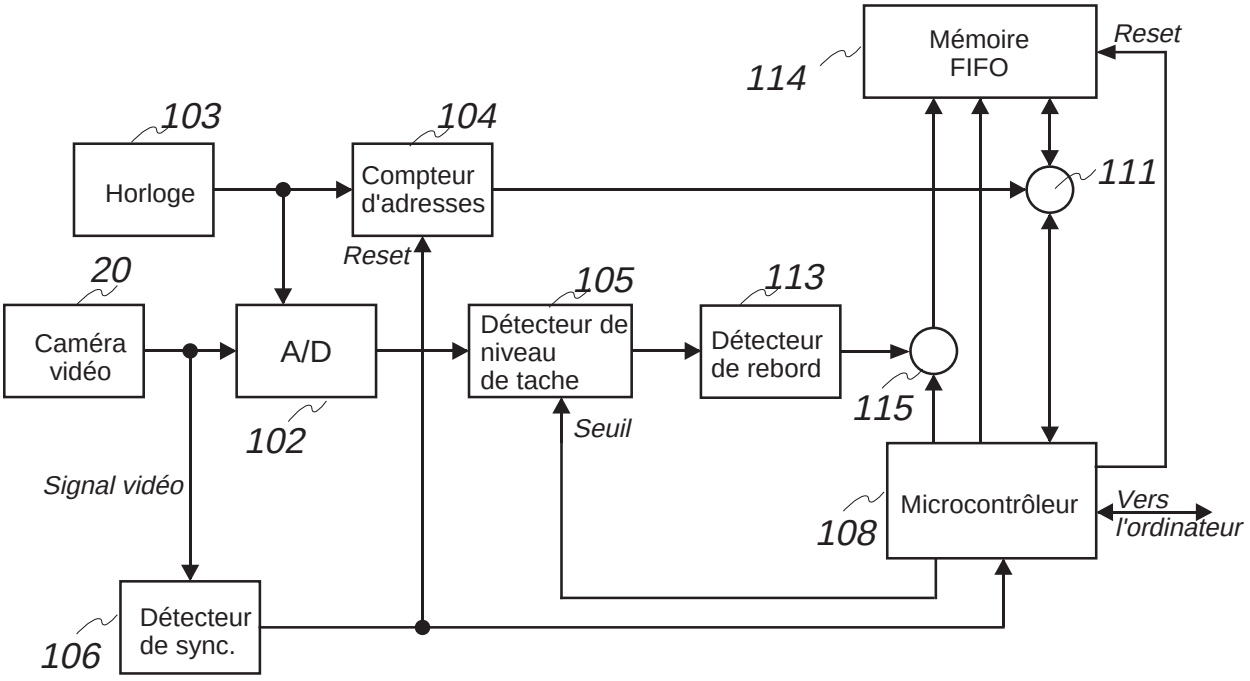

Figure 7.11: Unité de détection des centres de taches claires utilisant une mémoire FIFO.

Des composants supplémentaires sont nécessaires. Un détecteur de rebord de signal 113 est inséré en aval du détecteur de niveau de tache 105. Un sélecteur d'activation d'écriture 115 sélectionne des impulsions d'activation d'écriture venant soit du détecteur de rebord 113, soit du microcontrôleur 108. La sortie du compteur d'adresses 104 est connectée au sélecteur de circuit de données 111. En effet, on utilise la mémoire FIFO pour stocker uniquement les *adresses* des rebords montants et descendants des taches claires.

La figure 7.12 explique le mode d'opération. La barre en haut de la figure représente la sortie du détecteur de niveau de tache 105, et les bandes blanches et grises représentent respectivement les pixels de niveau haut et de niveau bas. Le détecteur de rebord 113 produit une impulsion à l'instant de la transition entre ces niveaux. Cette impulsion de rebord, par l'intermédiaire du sélecteur d'activation d'écriture 115, permet à la mémoire FIFO 114 d'enregistrer les adresses de colonne en provenance du compteur d'adresses 104 par l'intermédiaire du sélecteur du circuit des données 111 quand un rebord du signal se produit; ces adresses sont appelées a_1 et a_2 sur la figure 7.12. Dans la mémoire FIFO 114,

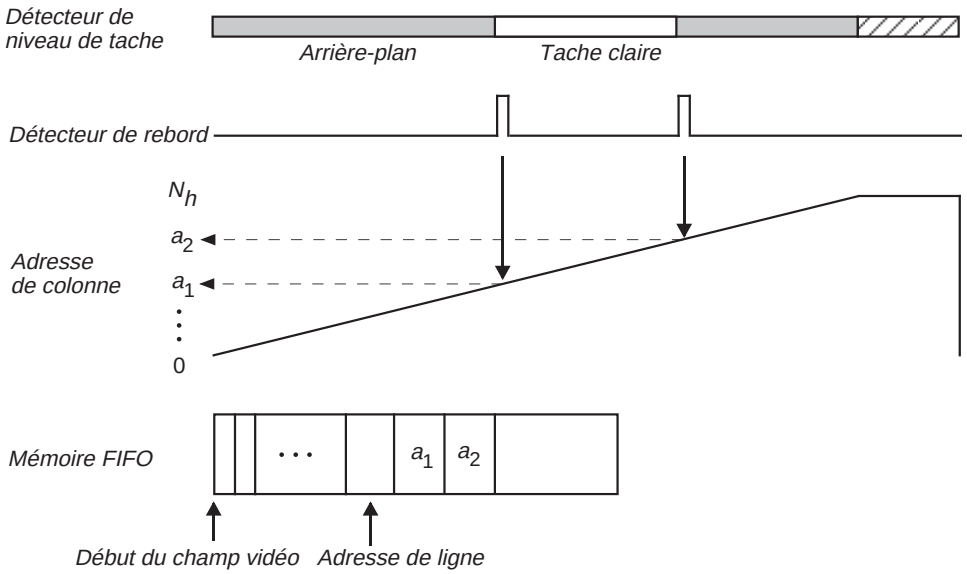

Figure 7.12: Enregistrement des adresses des rebords des taches claires dans la mémoire FIFO.

les données sont enregistrées séquentiellement à partir de l'adresse zéro, étant décalées par l'incrémentation du compteur interne. Le microcontrôleur réinitialise ce compteur à zéro au début de chaque trame d'image vidéo. Le bas de la figure 7.12 montre le contenu de la mémoire FIFO. Dans cet exemple, on suppose que le compteur d'adresses 104 génère seulement l'adresse de colonne; l'adresse de ligne est générée par le microcontrôleur pour spécifier complètement la position du rebord du signal. Durant la période de retour de balayage vertical, le microcontrôleur génère un signal d'activation de lecture, et lit toutes les adresses des rebords des taches claires par l'intermédiaire du sélecteur du circuit des données 111. Le microcontrôleur utilise ces adresses pour calculer la position des centres des taches.

Au lieu d'attendre la période de retour de balayage vertical, un microcontrôleur rapide peut utiliser les 10 μs de retour de balayage *horizontal* du signal vidéo pour lire les données de la mémoire FIFO, les transférer dans sa mémoire interne, et compléter les calculs nécessaires pour obtenir les centres des taches claires. Il n'a pas besoin dans ce cas d'envoyer des adresses de ligne à la mémoire FIFO. La mémoire FIFO est alors utilisée comme une mémoire tampon. Du fait du petit nombre de rebords de taches claires pour chaque ligne, quelques registres à décalages peuvent être utilisés pour jouer le rôle de mémoire FIFO.

7.6 En résumé

Nous avons présenté deux architectures pour la détection de taches claires dans le signal provenant d'une caméra vidéo. Ces architectures permettent d'opérer cette détection "en temps réel", c'est-à-dire toutes les $1/60^e$ de seconde, dans chaque trame du signal vidéo, avec des composants très peu coûteux. La première difficulté est que si on digitalise les images de façon à obtenir 242 lignes contenant chacune 256 pixels, un nouveau pixel est créé toutes les 200 νs, beaucoup trop vite pour qu'un microcontrôleur puisse intercepter ces pixels et les regrouper. On doit donc mettre en mémoire les données en attendant d'avoir le temps de les traiter. La deuxième difficulté est que la lecture des données devient alors un goulot d'étranglement. Dans la première architecture, on met en mémoire tous les pixels, tout en marquant les lignes sans pixels clairs. Le microcontrôleur profite du répit dans le signal vidéo à la fin de la transmission d'une trame pour lire les pixels clairs, et saute les lignes qui sont marquées vides. Dans la deuxième solution, on ne met en mémoire que les positions pour lesquelles les pixels passent du niveau bas au niveau haut et vice-versa. Le microcontrôleur profite du répit à la fin de la transmission de chaque ligne d'image pour lire les positions des rebords de tache claire pour cette ligne, et pour combiner cette information avec l'information acquise aux lignes précédentes, afin de trouver les centres des taches claires.

Chapitre 8

Logiciel pour un Traqueur Optique d'Interface 3D

Dans ce chapitre, nous décrivons les composants *logiciels* nécessaires pour le fonctionnement d'un traqueur optique permettant l'interaction avec une scène d'objets tridimensionnels représentés par un ordinateur. Nous décrivons le logiciel qui contrôle le système que nous avons construit, ainsi que d'autres solutions possibles que nous avons testées partiellement.

8.1 Pointeur, curseur virtuel et curseur d'écran

Sur la figure 8.1, un *curseur d'écran* est représenté sur l'image d'ordinateur, parmi des vues perspectives d'objets 3D – une clavette à section rectangulaire et un bloc cubique avec un trou à section rectangulaire. Le curseur d'écran est obtenu par la projection perspective d'un *curseur virtuel 3D*. Le curseur virtuel 3D est défini par exemple par une liste de coordonnées de points, et des listes d'indices spécifiant quels points appartiennent aux mêmes lignes, aux mêmes polygones. Le niveau de détail dans la description du curseur virtuel 3D est choisi en fonction des capacités graphiques de l'ordinateur. On suppose que cette structure géométrique est placée dans l'espace dans une position qui reflète la position du *pointeur 3D*

manipulé par l'opérateur (mais la section 8.3 montre qu'il n'est généralement pas souhaitable de faire coïncider la position du curseur virtuel et la position du pointeur 3D).

Le curseur d'écran est obtenu à partir du curseur virtuel 3D par la même transformation perspective utilisée pour projeter les autres objets de la scène virtuelle 3D sur l'écran d'ordinateur. Cette opération est plus précisément décrite dans la section 8.2.

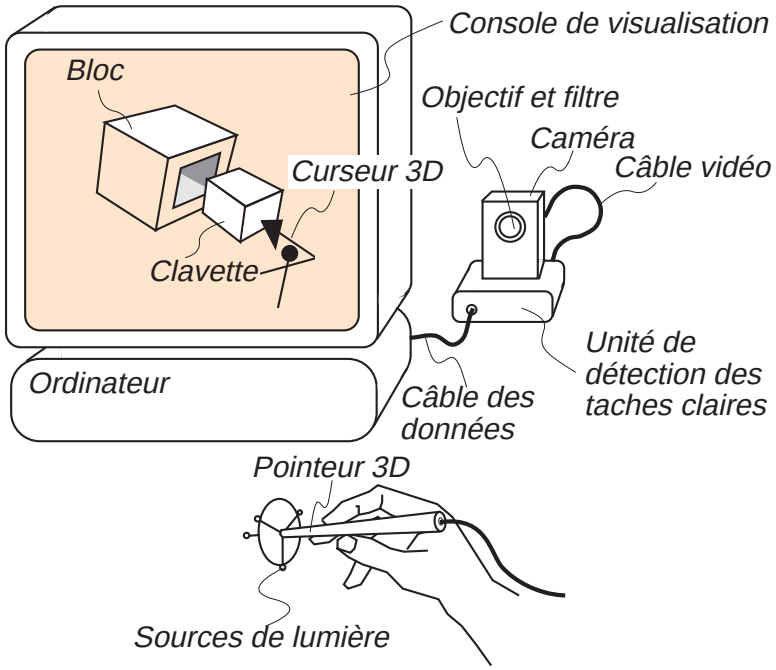

Figure 8.1: Eléments d'un système utilisant la vision artificielle pour un pointeur 3D.

Sur l'écran de l'ordinateur de la figure 8.1, le curseur virtuel 3D est un bonhomme avec une tête sphérique qui tient dans sa main droite une flèche perpendiculaire au plan de son corps et pointe cette flèche devant lui. L'opérateur a attaché le curseur à la clavette, et est en train d'insérer cette clavette dans le trou rectangulaire du bloc cubique. Le curseur peut constituer tour à tour un *outil* de manipulation, de création et de transformation de l'espace représenté sur l'écran; cet outil peut être choisi à partir d'une palette, une boîte à outils, un présentoir cylindrique tournant, etc...; la géométrie 3D du curseur devrait être suffisamment expressive et spécifique pour fournir à l'opérateur un rappel visuel de l'outil choisi ou de sa fonction, dans la tradition des gommes, pinceaux et pots de peinture des programmes de dessin 2D. Par exemple, dans un programme de sculpture pour enfants, la boîte à outils pourrait comprendre une baguette magique pour faire apparaître une boule de glaise, une

pompe pour augmenter le volume de la boule, une épingle pour diminuer le volume, une ventouse pour étirer ou repousser certaines parties, une perceuse pour faire des trous ronds (carrés, triangulaires, ...), une truelle pour créer des surfaces plates, etc....(chapitre 5).

8.2 Passage de la scène virtuelle à l'écran

La figure 8.2 illustre les opérations qui sont nécessaires pour générer un curseur sur l'écran à partir de la pose du pointeur 3D calculée par rapport à la caméra et de sa géométrie. Examinons d'abord comment on obtient les vues perspectives des objets 3D représentés sur l'écran d'ordinateur, tels que la clavette et le bloc cubique de la figure 8.1:

En l'absence de traqueur pour la tête de l'opérateur et de lunettes stéréoscopiques, le point de vue de l'opérateur doit être défini de manière un peu arbitraire; nous choisissons par exemple un point à une distance d'environ 1 m devant la caméra; un plan d'image pour l'opérateur est défini à une distance d'environ 20 cm devant le point de vue de l'opérateur, dans une position perpendiculaire à la ligne qui joint la caméra au point de vue de l'opérateur. Lorsqu'un objet virtuel doit être représenté, tel que le bloc cubique représenté sur la figure 8.1, cet objet virtuel est positionné dans l'espace situé devant l'opérateur. Une projection perspective de cet objet est calculée par la construction classique qui consiste à trouver les intersections de lignes de vue – joignant le point de vue de l'opérateur à des points caractéristiques de l'objet virtuel tels que les sommets du bloc cubique – avec le plan d'image de l'opérateur. Cette projection perspective est représentée sur l'écran d'ordinateur. Des effets de couleur et d'ombre sur les polygones joignant les points projetés sont calculés et représentés en fonction des orientations de ces surfaces pour fournir à l'opérateur une impression plus réaliste.

Avec un système stéréographique (lunettes à obturation alternée ou masque de visualisation), un point de vue et un plan d'image sont définis pour chacun des deux yeux de l'opérateur. Si de plus l'opérateur porte un traqueur sur ses lunettes ou son masque, ce traqueur fournit la position de la tête, à partir de laquelle on peut *calculer* dans l'espace les positions des points de vue et des plans d'image pour chaque œil, au lieu de choisir des positions arbitraires fixes. Les vues perspectives de la scène pour chaque œil sont alors calculées en utilisant ces points de vue et ces plans d'image mobiles.

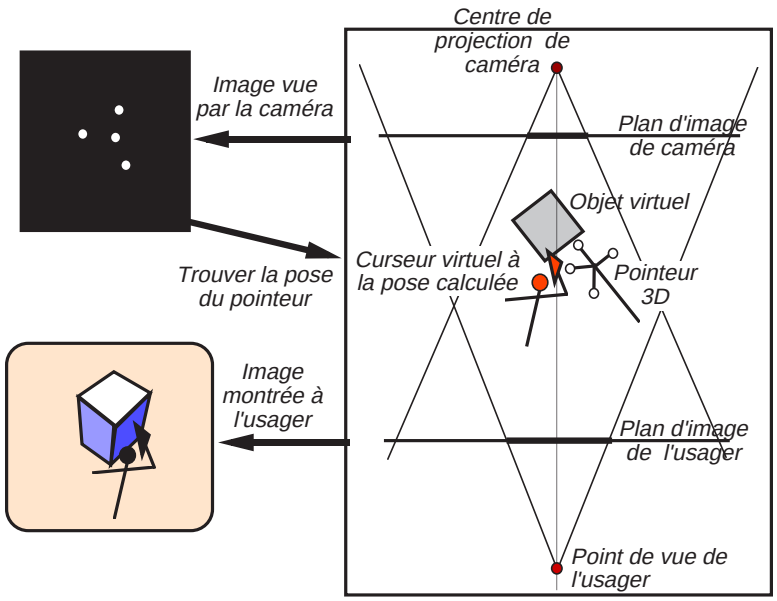

Figure 8.2: Passage de l'image des sources du pointeur à la représentation du curseur d'écran.

La figure 8.2 montre aussi la position réelle du pointeur 3D que l'opérateur a placé dans l'espace situé devant lui. La caméra fait face à l'opérateur, et l'image seuillée vue par la caméra est représentée à gauche de la figure. Les taches lumineuses qui sont les images des sources de lumière du pointeur 3D apparaissent sur un fond noir. Le fond est noir dans l'image grâce au filtre passe-bande placé devant la lentille de la caméra, et grâce à la sensibilité spécifique du capteur CCD de la caméra aux longueurs d'onde émises par les sources de lumière du pointeur (chapitre 7). De plus, une opération de seuillage est effectuée dans l'unité de détection des taches lumineuses. Cette unité a pour fonction de détecter les coordonnées des centres des taches lumineuses. A partir de ces coordonnées et de la connaissance de la configuration géométrique des sources sur le pointeur, le système calcule la pose du pointeur dans l'espace (chapitre 2). Le système peut alors positionner dans l'espace un curseur virtuel dans une position identique à celle du pointeur, comme il est montré sur la figure 3. Le système peut ensuite calculer les transformations perspectives de ce curseur virtuel par exactement les mêmes opérations utilisées pour calculer les transformations perspectives du bloc cubique virtuel.

De plus, le système peut vérifier si le curseur virtuel entre en collision avec le bloc virtuel, par exemple en calculant si un point spécifique du curseur virtuel est à l'intérieur du volume

géométrique qui définit le bloc. Si c'est le cas, l'opérateur peut exprimer le désir de déplacer le bloc cubique virtuel, par exemple en pressant un bouton situé sur le corps du pointeur ou sur le clavier de l'ordinateur. Le cube virtuel est alors connecté au pointeur 3D, et cette impression est donnée à l'opérateur par le fait que chaque changement de position du pointeur a pour résultat de produire un changement similaire de position du bloc cubique. Ce changement de position du bloc se traduit par un déplacement de la vue perspective du bloc sur l'écran d'ordinateur, qui donne à l'opérateur l'illusion qu'il est en train de déplacer le bloc 3D avec son pointeur.

8.3 Champ de vue de caméra et limites de l'écran

Quand l'opérateur déplace le pointeur en dehors du champ de vue de la caméra, une ou plusieurs taches lumineuses disparaissent de l'image, et le système ne peut plus calculer la position du pointeur. En conséquence, le curseur d'écran ne peut pas être redessiné en réponse aux mouvements de l'opérateur. Nous appelons cet événement *événement hors-caméra* dans la suite. Il est souhaitable que l'opérateur puisse détecter simplement en regardant l'écran quand il est sur le point de déplacer son pointeur 3D hors du champ de la caméra, de façon à éviter ces situations. Sur la figure 8.2, le champ de vue de la caméra est la région pyramidale de l'espace définie par le centre de projection de la caméra, sa distance focale, et le rectangle de la surface sensible du capteur CCD. Un événement hors-caméra se produit quand une au moins des sources lumineuses du pointeur sort des limites de cette région pyramidale.

De manière similaire, le champ de vue de l'opérateur est la région pyramidale définie par le point de vue de l'opérateur, la distance focale de l'opérateur, et le rectangle de vue qui correspond généralement à une *fenêtre sur l'écran*; dans la suite on suppose que cette fenêtre occupe tout l'écran. Lorsque le pointeur traverse une limite du champ de vue de l'opérateur, une partie du curseur d'écran disparaît hors de la fenêtre de vue. Nous appelons cet événement *événement hors-écran*. Les événements hors-écran ne créent pas de problèmes de calcul de pose, et sont facilement évitables puisque l'opérateur peut déplacer son pointeur 3D de façon à maintenir le curseur d'écran dans les limites de l'écran.

Le point important à noter est que si on fait simplement coïncider la pose du curseur

virtuel avec la pose calculée pour le pointeur 3D dans l'espace, les événements hors-caméra ne se produisent généralement pas en même temps que les événements hors-écran: Quand le pointeur 3D est proche de la caméra, il atteint une limite du champ de vue de la caméra bien avant d'atteindre une limite du champ de vue de l'opérateur, et l'opérateur qui fixe ses yeux sur l'écran est surpris de constater qu'il ne peut pas déplacer le curseur davantage, bien que le curseur soit encore loin des limites de l'écran. Par contre, quand le pointeur est loin de la caméra, il atteint une limite du champ de vue de l'opérateur bien avant d'atteindre une limite du champ de vue de la caméra. Dans ce cas, le curseur d'écran atteint le bord de l'écran alors que l'opérateur aurait pu continuer son mouvement sans problème.

Une solution à ces problèmes consiste à faire coïncider les événements hors-caméra et hors-écran, pour que l'opérateur reçoive un avertissement direct de l'imminence d'un événement hors-caméra en voyant que le curseur d'écran approche les limites de l'écran. Cet effet est obtenu par la multiplication de la translation latérale du curseur 3D par rapport à la translation réelle du pointeur 3D, pour que le curseur virtuel atteigne la limite du champ de vue de l'opérateur quand le pointeur atteint la limite du champ de caméra. La coïncidence entre le pointeur 3D et le curseur virtuel 3D est abandonnée. La translation latérale du curseur virtuel est amplifiée par rapport à la translation du pointeur 3D lorsque le pointeur est proche de la caméra, et réduite par rapport à la translation du pointeur 3D lorsque le pointeur 3D est loin de la caméra. Cette transformation de translation a l'avantage de laisser l'opérateur interagir avec les objets virtuels situés dans tout l'espace virtuel de son champ de vue, bien que ses mouvements réels soient en fait limités à l'espace situé dans le champ de vue de la caméra.

Les expériences montrent que l'accélération des déplacements latéraux de translation du curseur lorsque le pointeur est plus proche de la caméra est un problème bien moindre que le problème résolu par cette solution, et n'est souvent pas même remarquée par un utilisateur novice.

Dans certaines applications, il peut être avantageux de représenter la scène à l'intérieur d'une boîte, d'un aquarium, ou d'une scène de théâtre, avec des plans latéraux et de fond de scène qui limitent le champ visuel. Dans ce cas, lorque l'opérateur déplace son curseur au–delà de ces limites, le curseur disparaît ou refuse d'aller plus loin. On peut appeler un

tel événement *événement hors-boîte*. Il est également facile de faire coïncider les événements hors-boîte et hors-caméra en utilisant comme translations du curseur virtuel des transformations linéaires des translations du pointeur 3D, pour que le curseur atteigne les limites de la boîte quand le pointeur atteint les limites du champ de vue de la caméra.

8.4 Une géométrie simple pour les sources de lumière

Les figures 8.3 et 8.4 montrent une configuration géométrique simple des sources de lumière montées sur le pointeur. On se rappelle (chapitre 6) que cette géométrie est aussi préconisée pour un traqueur optique dans un brevet d'invention de Egli [17]. Dans ce brevet, les correspondances sont obtenues par des clignotements séquentiels des sources de lumière, et le calcul de pose utilise des équations du second degré. Notre dernier système fonctionne avec ce type de géométrie, mais utilise des sources constamment éclairées de manière à fournir un temps de réponse très court avec des caméras courantes (chapitre 6). Nous montrons dans cette section comment appliquer la méthode de calcul de pose présentée au chapitre 2, et nous examinons les avantages et inconvénients apportés par ce choix de géométrie.

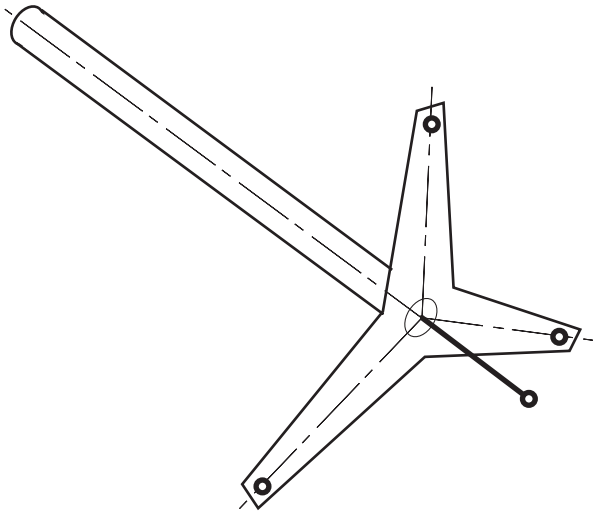

Figure 8.3: Pointeur symétrique à quatre sources de lumière.

L'armature du pointeur comporte quatre sources lumineuses – lampes à incandescence miniatures, diodes luminescentes, ou extrémités de fibres optiques produisant la lumière que doit détecter la caméra (voir chapitre 7). Ce nombre de sources est le nombre minimal requis

pour le calcul de pose décrit dans le chapitre 2. Le fait de choisir un nombre de sources égal au nombre minimal présente l'avantage d'augmenter les chances de détecter toutes les taches lumineuses correspondantes dans l'image vidéo; en effet, plus le nombre de sources est grand, plus le nombre de taches est grand, et plus est grand le risque d'avoir des taches superposées ou des sources occluses par une armature du pointeur. Un autre avantage d'un nombre réduit de sources est une réduction de la complexité du problème qui consiste à trouver la correspondance entre chaque tache lumineuse et la source qui l'a créée. Il y a $N!$ façons de mettre en correspondance N sources et N taches; ce nombre est seulement 24 pour 4 sources, mais 120 pour 5, 720 pour 6, etc.... En fait, du fait des symétries de l'arrangement des sources dans la figure 8.3, on n'a pas à considérer 24 possibilités, mais uniquement 10, comme on le voit dans la suite. Parmi les inconvénients, nous allons voir que du fait du petit nombre de sources et des symétries, les images sont généralement ambiguës, et pour chaque image, il est difficile dans certains cas de choisir la pose correcte parmi les poses possibles (section 8.7).

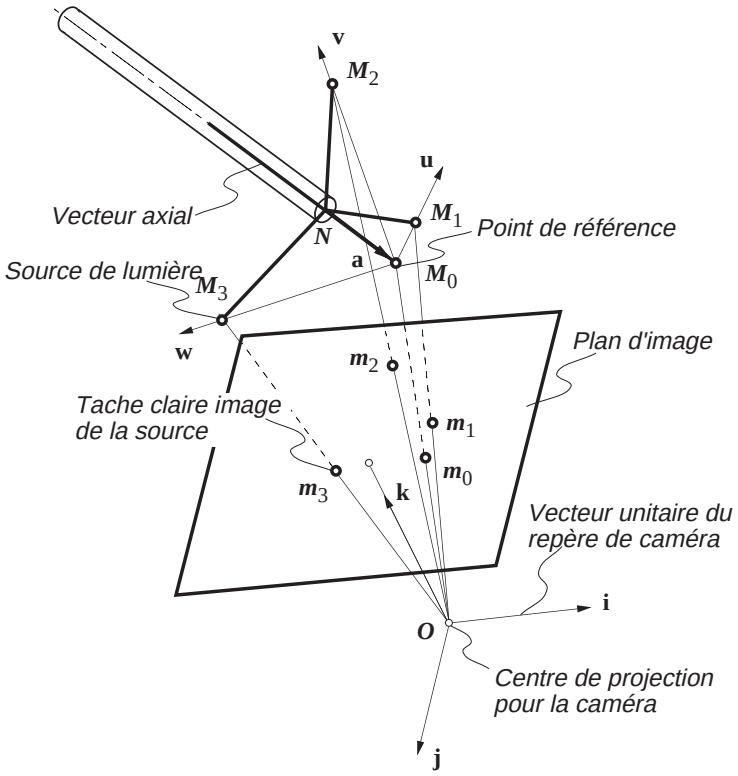

Figure 8.4: Notations pour le pointeur à quatre sources.

8.5 Notations pour le pointeur à quatre sources

La figure 8.4 introduit les notations nécessaires pour les explications qui suivent. Nous réutilisons quand c'est possible les notations introduites sur la figure 2.1. La source qui est située sur l'axe du pointeur est appelée *source de référence*, et se trouve au point M_0, lui-même appelé point de référence du pointeur. Ce point M_0 est l'origine d'un repère cartésien dénoté $(M_0\mathbf{u}, M_0\mathbf{v}, M_0\mathbf{w})$, choisi comme repère de référence pour le pointeur. La deuxième source est située au point M_1 sur l'axe des u à la coordonnée $u = 1$, la troisième source est située au point M_2 sur l'axe des v à la coordonnée $v = 1$, et la quatrième source est située au point M_3 sur l'axe des w à la coordonnée $w = 1$. Un vecteur $\mathbf{a}$ est défini le long de l'axe du pointeur, et est appelé le *vecteur axial du pointeur*. Ce vecteur peut être calculé par l'expression

$$\mathbf{a} = -(\mathbf{M_0M_1} + \mathbf{M_0M_2} + \mathbf{M_0M_3})$$

Le signe $-$ est introduit de façon que le vecteur $\mathbf{a}$ soit dirigé du corps du pointeur vers le point de référence, c'est-à-dire dans la direction dans laquelle pointe le pointeur. Ce vecteur est utilisé pour vérifier si le pointeur pointe vers la caméra ou dans la direction opposée, et est utilisé en cas d'ambiguïté. Le point N est défini comme l'intersection de l'axe du pointeur avec le plan contenant les points M_1, M_2 et M_3. Ce plan est perpendiculaire au vecteur axial $\mathbf{a}$. Les images des sources sur le plan d'image sont des taches de lumière dont les centres sont notés m_0, m_1, m_2, m_3. Le repère cartésien de la caméra a comme vecteurs unitaires les vecteurs $\mathbf{i}$, $\mathbf{j}$ parallèles au plan de l'image, et $\mathbf{k}$ pointant dans une direction perpendiculaire à ce plan, le long de l'axe optique de la caméra.

8.6 Calcul de pose pour le pointeur à quatre sources

Cette géométrie des sources simplifie le calcul de pose du pointeur par l'algorithme POSIT (chapitre 2). D'abord, la matrice $\mathbf{A}$ des équations 2.12, dont les lignes sont les coordonnées des sources dans le repère du pointeur, est ici simplement une matrice identité 3×3. La matrice d'objet $\mathbf{B}$ du pointeur, qui est la matrice inverse de A lorsqu'on considére quatre

points (équations 2.13), est donc aussi une matrice identité 3×3. Les centres des taches de lumière, m_0, m_1, m_2, m_3 sont détectés par l'unité de détection des taches (voir chapitre 7), qui produit leurs coordonnées $(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$ dans l'image. Le calcul de pose fait appel aux vecteurs images $\mathbf{x}'$ et $\mathbf{y}'$, définis par

$$\mathbf{x}' = (\xi_1, \xi_2, \xi_3), \quad \mathbf{y}' = (\eta_1, \eta_2, \eta_3),$$

avec $\xi_i = x_i(1 + \varepsilon_i) - x_0, \eta_i = y_i(1 + \varepsilon_i) - y_0$. Les termes ε_i sont égaux à

$$\varepsilon_i = \frac{1}{Z_0}\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{k}$$

Puisque la matrice d'objet $\mathbf{B}$ est une matrice identité, on trouve pour les vecteurs $\mathbf{I}$ et $\mathbf{J}$:

$$\mathbf{I} = \mathbf{x}', \quad \mathbf{J} = \mathbf{y}'$$

On se rappelle que dans l'algorithme POSIT, la première ligne de la matrice de rotation est ensuite calculée en normalisant $\mathbf{I}$, la deuxième ligne en normalisant $\mathbf{J}$. Les deux premières lignes de cette matrice sont donc $\mathbf{i} = \mathbf{I}/|\mathbf{I}|$, $\mathbf{j} = \mathbf{J}/|\mathbf{J}|$. La troisième ligne $\mathbf{k}$ est le produit vectoriel des deux premières lignes. On peut utiliser pour le calcul des coordonnées de $\mathbf{I}$ et $\mathbf{J}$ les valeurs des quantités ε_i correspondant à la pose calculée pour l'image vidéo précédente, si cette pose est disponible. Pour la première image, on peut soit se contenter d'une pose approximative en posant $\varepsilon_i = 0$, soit améliorer la précision de la pose par quelques itérations. On se rappelle qu'une itération a pour base le fait que, une fois qu'on a calculé la troisième ligne correspondant au vecteur $\mathbf{k}$ et la coordonnée Z_0 de la translation, on peut calculer des valeurs plus précises pour ε_i. Le processus d'itération peut être évité pour un pointeur 3D qui ne requiert pas une grande précision, mais est important pour un traqueur de tête.

Après le calcul de la matrice de rotation (avec ou sans itérations), le vecteur de translation est obtenu en multipliant le vecteur Om_0 par la distance focale de la caméra et par l'inverse de la norme de $\mathbf{I}$ ou $\mathbf{J}$.

Cette méthode suppose qu'on a trouvé les correspondances entre sources et taches de lumière. Ce fait devient plus clair dans la section suivante, où nous montrons ce qui se passe en cas d'erreur de correspondance, et décrivons un procédé pour trouver les correspondances correctes.

8.7 Correspondances pour le pointeur à quatre sources

8.7.1 Détection analytique des correspondances

Tant que les correspondances sont inconnues, on ne peut calculer les quantités telles que $\xi_i = x_i(1+\varepsilon_i) - x_0$. En effet, les termes ε_i dépendent des vecteurs $\mathbf{M_0M}_i$, tandis que les termes x_i dépendent des images m_i. On ne peut donc apparier x_i à $(1+\varepsilon_i)$ pour calculer leur produit que si on connaît les correspondances entre M_i et m_i.

Mais les projections orthogonales à l'échelle des sources de lumière (voir chapitre 2) sont généralement proches des taches claires actuelles produites par projection perspective. Nous nous servons de cette approximation pour trouver les correspondances entre sources de lumière et taches claires. Cette approximation revient à supposer que les quantités ε_i sont nulles dans les équations ci–dessus, c'est-à-dire que les vecteurs $\mathbf{I}$ et $\mathbf{J}$ sont approximativement égaux à

$$\mathbf{I} = (x'_1, x'_2, x'_3), \quad \mathbf{J}' = (y'_1, y'_2, y'_3),$$

avec $x'_i = x_i - x_0, y'_i = y_i \ - y_0$.

Dans la section 2.13, nous avons introduit une "mesure d'erreur orthonormale", et mentionné qu'on peut utiliser cette mesure pour détecter les mises en correspondance incorrectes entre points d'objet et points d'image. Cette mesure utilise le fait d'une part que les deux vecteurs $\mathbf{I}$ et $\mathbf{J}$ doivent être orthogonaux, d'autre part que ces deux vecteurs doivent être de même module. Examinons ces deux conditions séparement dans le cas particulier du pointeur à quatre points: La première condition s'écrit $\mathbf{I} \cdot \mathbf{J} = 0$, c'est-à-dire dans ce cas

$$C_1 = x'_1 y'_1 + x'_2 y'_2 + x'_3 y'_3 = 0$$

La deuxième condition s'écrit $\mathbf{I}^2 - \mathbf{J}^2 = 0$, c'est-à-dire dans ce cas

$$C_2 = x'^2_1 + x'^2_2 + x'^2_3 - y'^2_1 - y'^2_2 - y'^2_3 = 0$$

Si la géométrie des quatre sources sur le pointeur était quelconque, on devrait calculer les mesures C_1 et C_2 pour les 24 correspondances possibles entre sources et taches claires,

et choisir comme correspondance celle qui correspond aux plus petites valeurs de C_1 et C_2. Toutefois, avec la configuration symétrique des sources illustrée sur la figure 8.3, nous montrons dans les paragraphes qui suivent que le nombre de tests peut être réduit. Plus précisément, il suffit de quatre tests pour sélectionner parmi les quatre taches de lumière celle qui correspond au point de référence M_0. Ensuite, six tests sont nécessaires pour trouver les correspondances pour les points M_1, M_2, et M_3. Nous montrons que les mesures C_1 et C_2 ne sont pas utiles dans ces six derniers tests, à cause des symétries de rotation des sources. Ces tests doivent comparer la matrice de rotation et le vecteur axial pour l'image vidéo considerée avec la matrice de rotation et le vecteur axial obtenus $1/60^e$ s plus tôt pour l'image précédente.

8.7.2 Ambiguïtés de rotation et de symétrie

Dans la configuration des sources illustrée sur la figure 8.4, une rotation de 120 degrés du pointeur autour de son axe garde en place la source M_0, et déplace M_1 en M_2, M_2 en M_3, et M_3 en M_1. L'image du pointeur après la rotation est exactement la même qu'avant la rotation. Une rotation opposée de -120 degrés produit aussi la même image. Par conséquent, étant donné quatre taches de lumière, il y a (au moins) trois matrices de rotation possibles qui peuvent être trouvées pour le pointeur. Nous rappelons que dans la matrice de rotation les colonnes sont les coordonnées des vecteurs unitaires du repère cartésien du pointeur dans le repère de la caméra. Ces vecteurs unitaires sont les vecteurs $\mathbf{M_0M_1}$, $\mathbf{M_0M_2}$ et $\mathbf{M_0M_3}$. Une rotation de 120 ou -120 degrés revient à la permutation circulaire des noms de ces vecteurs. La matrice de rotation correspondante est donc obtenue par une permutation circulaire correspondante des colonnes de la matrice de rotation. On ne peut pas distinguer dans l'image laquelle de ces trois rotations est correcte. Les mesures C_1 et C_2 ne peuvent pas non plus nous aider dans cette différenciation: en effet les expressions de ces deux mesures sont invariantes dans les permutations circulaires des coordonnées des points d'image m_1, m_2 et m_3.

Il existe une autre ambiguïté de correspondance plus subtile avec cette géométrie, illustrée sur la figure 8.5: Si on permute simplement les labels de deux points d'image tels que m_1

et m_2, les conditions C_1 et C_2 restent inchangées, ce qui montre qu'on obtient encore deux vecteurs **I** et **J** perpendiculaires et de même module. La pose correspondante est donc encore acceptable, mais tout à fait différente de la pose obtenue avant la permutation. Sur la figure 8.5, la première pose est illustrée à gauche, et la deuxième pose acceptable, correspondant à des points d'image à la même position mais avec les labels m_2 et m_3 permutés, est illustrée à droite.

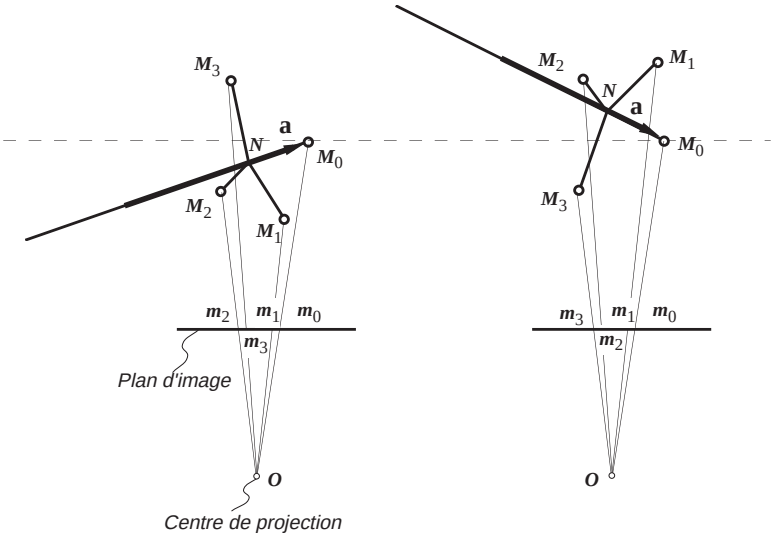

Figure 8.5: Deux poses symétriques donnant la même image avec des correspondances permutées pour les points M_2 et M_3.

Pour comprendre la relation entre ces deux poses, il faut se rappeler que la troisième ligne de la matrice de rotation est obtenue par le produit vectoriel des deux premières lignes. La troisième ligne est donc un vecteur proportionnel au produit vectoriel de **I** et **J**. Les vecteurs **I** et **J** ont pour coordonnées (x'_1, x'_2, x'_3) et (y'_1, y'_2, y'_3) avant la permutation, et (x'_1, x'_3, x'_2) et (y'_1, y'_3, y'_2) lorsque les labels m_2 et m_3 sont permutés. La troisième ligne de la matrice de rotation est proportionnelle à $(x'_2y'_3 - x'_3y'_2, x'_3y'_1 - x'_1y'_3, x'_1y'_2 - x'_2y'_1)$ avant la permutation, et à $(x'_3y'_2 - x'_2y'_3, x'_2y'_1 - x'_1y'_2, x'_1y'_3 - x'_3y'_1)$ une fois que m_2 et m_3 ont été permutés. D'autre part, on sait que les éléments de cette troisième ligne sont les coordonnées en z des vecteurs $\mathbf{M_0M_1}$, $\mathbf{M_0M_2}$ et $\mathbf{M_0M_3}$ dans le repère de la caméra. La permutation dans l'image a (1) permuté les coordonnées en z de M_2 et M_3 et (2) changé le signe de ces coordonnées en z. Ces changements indiquent que la pose correspondant à une image dans laquelle m_2 et m_3 sont permutés correspond à une pose dans laquelle M_2 et M_3 sont permutés, et dans laquelle

le pointeur est approximativement symétrique par rapport au plan parallèle au plan d'image passant par M_0 (figure 8.5).

Si deux poses correspondant à une permutation de deux points d'image sont dans cette relation géométrique, les *vecteurs axiaux* du pointeur pour ces deux poses ont des coordonnées en z de signes opposés. Ceci peut être vérifié directement:

Le vecteur axial $\mathbf{a}$ a des coordonnées dans le repère de la caméra qui sont proportionnelles à la somme des coordonnées des vecteurs $\mathbf{M_0M_1}$, $\mathbf{M_0M_2}$ et $\mathbf{M_0M_3}$. Donc les coordonnées du vecteur $\mathbf{a}$ sont respectivement proportionnelles aux sommes des termes dans les trois lignes de la matrice de rotation. Ces sommes sont

$$a_x = x'_1 + x'_2 + x'_3, \quad a_y = y'_1 + y'_2 + y'_3$$

$$a_z = x'_3y'_1 + x'_1y'_2 + x'_2y'_3 - x'_3y'_2 - x'_2y'_1 - x'_1y'_3$$

avant la permutation de m_2 et m_3, et

$$a_x = x'_1 + x'_3 + x'_2, \quad a_y = y'_1 + y'_3 + y'_2$$

$$a_z = x'_2y'_1 + x'_1y'_3 + x'_3y'_2 - x'_2y'_3 - x'_3y'_1 - x'_1y'_2$$

après la permutation.

On voit que les coordonnées a_x et a_y du vecteur axial $\mathbf{a}$ sont indépendantes des permutations entre les labels des points images m_1, m_2 et m_3, tandis que la troisième coordonnée change de signe selon le signe de la permutation.

On note enfin que les coordonnées de $\mathbf{a}$ et les conditions C_1 et C_2 sont sensibles à la correspondance correcte entre le point image m_0 et le point de référence M_0, si on se rappelle que toutes les quantités x'_i et y'_i contiennent x_0 et y_0 dans leurs expressions.

8.7.3 Stratégie de mise en correspondance

La stratégie proposée pour déterminer la correspondance correcte entre les sources et leurs images pour le pointeur à quatre sources de la figure 8.3 découle directement de la discussion ci–dessus.

Dans un premier temps, l'image correcte m_0 du point M_0 est détectée. Dans ce but, le système calcule pour chacun des quatre choix d'images possibles les valeurs des deux mesures C_1, C_2, et des coordonnées a_x et a_y du vecteur axial **a**. Comme on l'a vu, ces calculs ne nécessitent pas de connaître la correspondance correcte pour les autres points m_1, m_2 et m_3. Une troisième mesure de correspondance C_3 est obtenue en combinant a_x, a_y, et les coordonnées a_{0x}, a_{0y} du vecteur axial **a** pour l'image précédente. Par exemple

$$C_3 = (a_x - a_{0x})^2 + (a_y - a_{0y})^2$$

Parmi les quatre choix pour m_0, le point d'image qui fournit la plus petite mesure totale $C = C_1 + C_2 + C_3$ est choisie.

Dans un deuxième temps, on cherche les autres correspondances; les coordonnées x_0 et y_0 de m_0 sont maintenant connues, et nous avons vu que les quantités C_1 et C_2 ne sont plus utiles pour trouver les autres correspondances. Nous considérons donc une correspondance *arbitraire* entre les trois points d'image restants et M_1, M_2 et M_3, et calculons les vecteurs **I** et **J**:

$$\mathbf{I} = (x'_1, x'_2, x'_3), \quad \mathbf{J} = (y'_1, y'_2, y'_3)$$

Supposant que les vecteurs $\mathbf{I}_0$ et $\mathbf{J}_0$ obtenus une fraction de seconde auparavant à partir de l'image précédente sont corrects, nous permutons simultanément l'ordre des coordonnées de **I** et **J**, et sélectionnons la permutation qui produit les vecteurs les plus proches de $\mathbf{I}_0$ et $\mathbf{J}_0$.

8.7.4 Limiter les mouvements pour limiter les ambiguïtés

Cette méthode doit être complétée par l'utilisation d'une contrainte supplémentaire, sans quoi elle serait incapable dans certaines situations de choisir entre deux poses distinctes du pointeur. Un exemple de ce type de situation est illustré sur les figures 8.6, (a), (b) et (c). La figure 8.6(a) décrit une pose du pointeur dans laquelle les points d'image m_2 et m_3 sont confondus parce que les sources M_2 et M_3 sont alignées sur la même ligne de vue. Pour que cela se produise, il faut que le vecteur **a** soit approximativement perpendiculaire à l'axe optique. Dans ce cas, le vecteur **I** a pour coordonnées $(x'_1, x'_2, x'_3 = x'_2)$. Dans l'image suivante, ces deux points d'image se sont séparés, et la figure 8.6(b) montre une interprétation possible de

la pose du pointeur pour cette image, pour laquelle **I** a pour coordonnées (xx'_1, xx'_2, xx'_3), où xx' est maintenant utilisé au lieu de x' pour indiquer que les coordonnées en x dans l'image ont changé. La figure 8.6(c) montre une seconde interprétation possible de la pose du pointeur pour cette image, dans laquelle les points d'image correspondant à M_2 et M_3 sont permutés, et pour laquelle **I** a pour coordonnées (xx'_1, xx'_3, xx'_2). Le changement de module du vecteur **I** entre les figures 17 (a) et 17 (b) est la racine carrée de $(xx'_1 - x'_1)^2 + (xx'_2 - x'_2)^2 + (xx'_3 - x'_2)^2$. Le changement de module du vecteur **I** entre les figures 8.6(a) et 8.6(c) est la racine carrée de $(xx'_1 - x'_1)^2 + (xx'_3 - x'_2)^2 + (xx'_2 - x'_2)^2$, c'est-à-dire exactement la même quantité. Par conséquent une comparaison du vecteur **I** – et de même **J** – entre des images successives n'est pas suffisante pour discriminer dans ce type de situation deux interprétations possibles de la même image. Mais ces deux interprétations possibles sont obtenues en permutant les labels des points d'image m_2 et m_3, et comme on l'a vu ci–dessus en référence à la figure 8.5, les deux poses correspondantes ont des vecteurs axiaux **a** qui sont symétriques par rapport à un plan parallèle au plan d'image et ont des coordonnées en z de signe opposé. Ceci est illustré par le fait que les projections h du vecteur **a** sur l'axe des z de la caméra ont des directions opposées sur les figures 8.6(b) et 8.6(c). Nous résolvons cette ambiguïté de pose en choisissant toujours la pose qui correspond au vecteur **a** pointant vers la caméra, et en rejetant l'autre solution possible. Pour cela on choisit toujours la pose telle que le signe de l'expression

$$a_z = x'_3 y'_1 + x'_1 y'_2 + x'_2 y'_3 - x'_3 y'_2 - x'_2 y'_1 - x'_1 y'_3$$

soit positif. Mais pour que ce choix corresponde à la réalité, il faut demander à l'opérateur qu'il restreigne les orientations du pointeur aux angles pour lesquels le vecteur axial pointe vers la caméra. C'est en fait une limitation raisonnable en pratique, parce que dès que l'opérateur oriente le pointeur dans la direction opposée à la caméra, il y a de grandes chances qu'il obstrue une des sources de lumière avec la main située sur la poignée, ou avec l'armature du pointeur. Lorsque la position du pointeur atteint les limites du domaine angulaire admissible, la quantité a_z devient très petite, et le système peut utiliser cette information pour fournir un avertissement sous forme graphique, par exemple par un changement de couleur ou d'apparence du curseur d'écran.

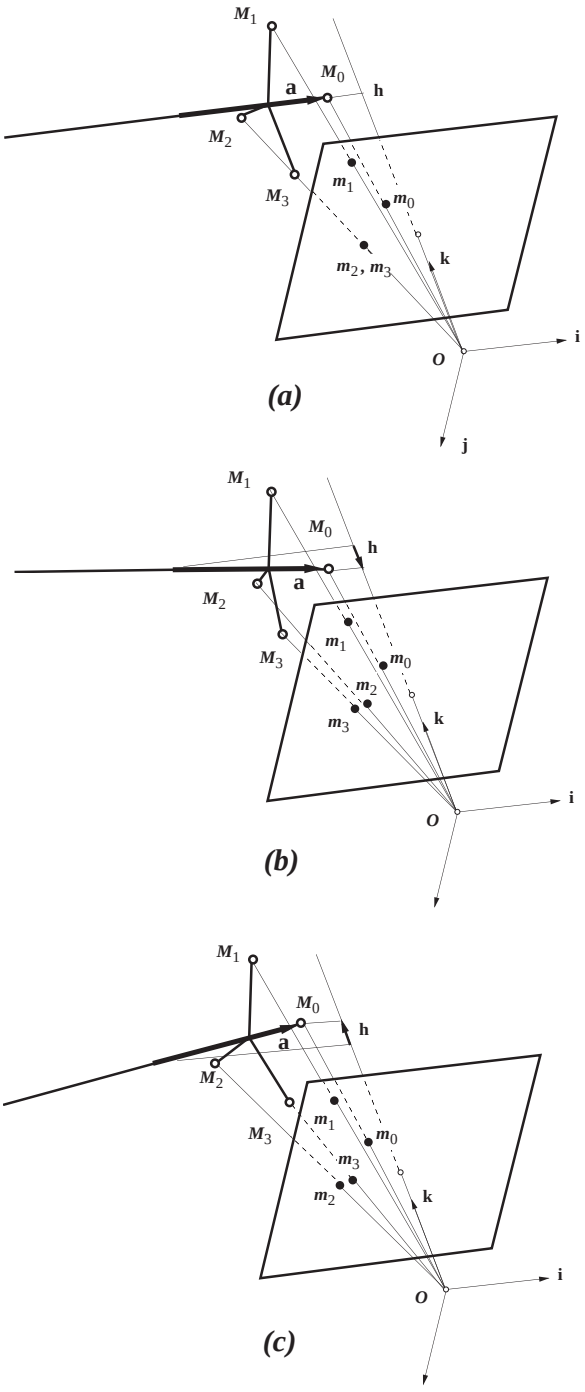

Figure 8.6: Ambiguïté de pose après la superposition des images des sources M_2 et M_3. En (a), ces deux points sont sur la même ligne de vue et leurs projections coïncident. Sur la figure (b), ces projections sont à nouveau séparées, mais on ne peut conclure si le pointeur est dans la pose illustrée en (b), pointant légèrement vers la caméra, ou dans celle illustrée en (c), pointant dans l'autre direction. On résout cette ambiguïté en supposant que le pointeur pointe toujours vers la caméra, et en avertissant l'opérateur quand l'axe du pointeur est sur le point d'être perpendiculaire à l'axe optique de la caméra.

8.7.5 Pose pour l'image de départ

Le calcul de la pose du pointeur par la méthode qui vient d'être décrite nécessite l'utilisation des vecteurs **I**, **J**, et **a** calculés pour l'image précédente. Par conséquent, une procédure légèrement différente est nécessaire au départ lorsqu'aucune image n'est encore disponible. Pour cela, il est demandé à l'utilisateur de démarrer le processus d'affichage du curseur en pointant le pointeur vers la caméra. Le système vérifie que l'opérateur est dans cette position en calculant l'angle entre le vecteur axial **a** du pointeur et l'axe optique de la caméra. Ce calcul nécessite de trouver le point d'image correspondant au point de référence M_0, une opération qui n'exige pas la connaissance d'une pose précédente si C_1 et C_2 seulement sont utilisés (et non C_3). Dès que l'angle devient suffisamment petit, le système peut supposer que l'opérateur a son pointeur en direction de la caméra, et peut calculer le vecteur axial **a**, et la matrice de rotation (parmi les trois interprétations possibles du fait de la symétrie du pointeur, on choisit arbitrairement celle pour laquelle la source située au–dessus des deux autres à ce moment est M_1). Le système peut alors utiliser cette pose initiale pour calculer une première pose réelle, qui pourra être utilisée pour la pose suivante, etc....

8.8 Disparition d'images de sources

8.8.1 Description du problème

Nous présentons maintenant les procédures utilisées quand moins de quatre taches de lumière sont détectées dans une image. La solution de facilité serait la suivante: puisque la pose du pointeur ne peut pas être calculée, le curseur reste immobile sur l'écran en attendant une nouvelle pose. Dès que quatre taches de lumière sont détectées à nouveau, le curseur est remis à jour. Cela peut produire un saut du curseur si l'opérateur a déplacé le pointeur sur une distance ou un angle non négligeable pendant la disparition des taches lumineuses. Cela crée un problème parce que l'opérateur voit un déplacement continu de sa main se traduire par un déplacement saccadé sur l'écran. Si la disparition de taches lumineuses dure un certain temps, un autre problème plus important peut apparaître: nous avons vu dans les sections précédentes que le calcul de pose utilise des quantités calculées dans l'image

précédente, et suppose que ces quantités n'ont pas beaucoup changé. Cette hypothèse est valide si les calculs sont effectués à un taux de 60 poses par seconde comme c'est le cas en fonctionnement normal, mais devient invalide si, d'une part, aucune pose n'a pu être calculée pour quelques images du fait de taches de lumière manquantes, et d'autre part la pose du pointeur a beaucoup changé pendant ce temps. Pour cette raison, il est préférable de renvoyer le système à son mode d'initialisation décrit ci–dessus s'il n'a pas pu trouver une pose pendant quelques secondes.

L'approche que nous avons prise pour minimiser le problème de disparition de taches consiste d'abord à éviter les causes du problème , ensuite à interpréter et déduire l'information manquante pour continuer à opérer.

8.8.2 Eviter les causes du problème

Les causes principales de disparition de taches de lumière sont les suivantes

1. Evénement hors-caméra
2. Occlusion par la main de l'opérateur
3. Occlusion par une armature du pointeur
4. Superposition de taches de lumière

Les événements hors-caméra sont évités par la technique décrite dans la section 8.3. On rappelle qu'avec cette technique la translation du curseur est modifiée par rapport à la translation réelle du pointeur pour qu'un événement hors-caméra coïncide avec un événement hors-écran. Voyant le curseur à la limite de l'écran (ou d'une fenêtre, ou d'une boîte dans laquelle la scène est représentée), l'opérateur sait que son pointeur est sur le point de sortir des limites du champ de la caméra et limite son mouvement.

Nous avons aussi vu ci–dessus (section 8.7.4) que pour éviter une ambiguïté d'intreprétation d'image, l'opérateur doit toujours pointer le pointeur vers la caméra. En d'autre termes, l'angle des rotations utilisables correspond à une demi-sphère, et non à une sphére complète. Quand le vecteur axial du pointeur devient presque perpendiculaire à l'axe optique

de la caméra, l'opérateur reçoit un signal graphique indiquant que la limite de rotation est atteinte, par exemple par un changement de couleur ou un clignotement du curseur. Dans ces conditions, l'occlusion par la main de l'opérateur ne peut pas se produire, car la main est toujours derrière le plan (M_1, M_2, M_3) par rapport à la caméra.

A cause de ces limitations de rotation du pointeur, le seul risque d'occlusion par une armature est dû à l'occlusion d'une source par le support NM_0 de la source de référence M_0 (figure 8.4). Ce support devrait donc être de préférence fin et transparent. Les autres occlusions d'armature ne sont possibles que lorsque le pointeur pointe dans la direction opposée à la caméra. Un support opaque en forme de disque pour les sources M_1, M_2 et M_3 ne devrait donc pas créer de problèmes.

Une superposition de taches de lumière a moins de chance de se produire si les taches ne couvrent qu'un ou deux pixels. On peut donc minimiser l'apparition de cet événement en combinant le choix de sources de lumière de petite dimension telles que les extrémités de fibres optiques (figure 7.3) avec un contrôle dynamique du seuil de détection des taches claires dans l'image [15].

8.8.3 Interpréter l'information manquante

On note qu'avec la géométrie proposée sur la figure 8.4 et les limitations de rotations à une demi-sphère, une superposition entre les images des points M_1, M_2 et M_3 ne peut pas se produire. On ne doit donc considérer comme probable que la superposition de l'image de la source M_0 avec l'image d'une seule des autres sources, M_1, M_2 ou M_3. Cet événement est illustré sur la figure 8.7, où les points M_0 et M_1 sont sur la même ligne de vue et les points d'image m_0 et m_1 coïncident. Le système est en fait capable de détecter un tel événement par la détection des deux événements suivants:

- Trois seulement des quatre points d'image sont détectés dans l'image.
- Dans l'image précédente, le point d'image m_0 était très proche d'un autre point d'image.

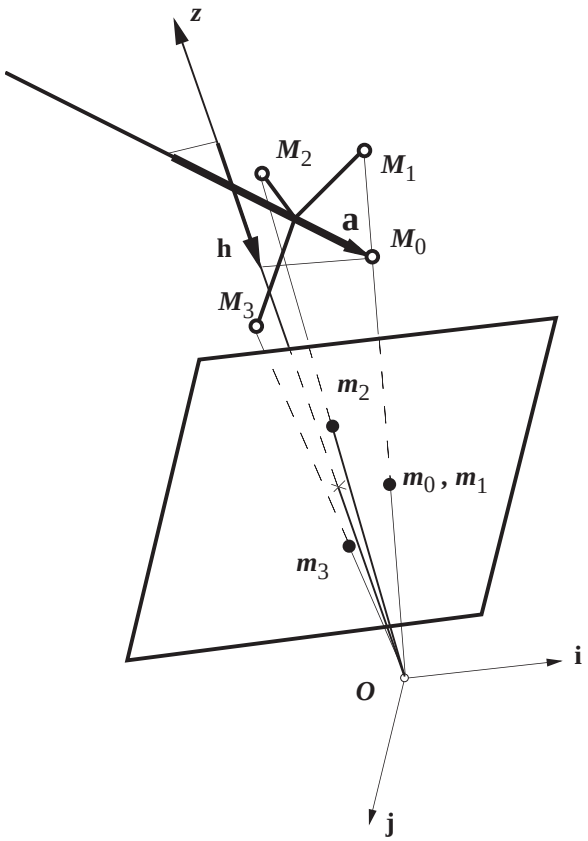

Figure 8.7: Superposition des images du point de référence M_0 et du point M_1.

Quand cette séquence se produit, le système peut supposer avec une probabilité élevée que le point m_0 s'est superposé avec un autre point dans l'image. Le système suit alors une stratégie similaire à celle du cas général; il trouve le point m_0 en testant successivement comme candidat chacun des trois points d'image disponibles, et pour chaque test il crée un quatrième point d'image qui coïncide avec m_0. Comme dans le cas général, il sélectionne comme point m_0 le point qui minimise la mesure $C = C_1 + C_2 + C_3$. Le reste du calcul se poursuit comme auparavant avec les trois autres points, ces points incluant le quatrième point d'image ajouté à la même position que m_0.

8.9 Autres méthodes

Les méthodes décrites ci–dessus sont plus simples à programmer qu'à expliquer; les programmes correspondants sont rapides, et donnent de bons résultats. Par exemple, par rapport à l'examen de toutes les correspondances possibles, la méthode de correspondance proposée est deux fois plus rapide. Toutefois, ce gain de vitesse est utile sur un Macintosh IIci, mais pas nécessairement sur une machine Silicon Graphics. On peut par exemple utiliser un pointeur 3D avec quatre sources de lumière sans symétries de rotation, et on calcule une combinaison linéaire C des mesures C_1, C_2 et C_3 définies ci–dessus pour les 24 combinaisons possibles entre sources et taches claires détectées. On retient la correspondance qui produit la plus petite valeur pour C. Mais avec un petit nombre de points, il est possible d'avoir de temps en temps des images ambiguës correspondant à plusieurs poses possibles. Il est donc prudent de comparer la pose trouvée avec la pose de l'image précédente et d'éliminer une pose trop différente.

Utiliser un plus grand nombre de sources de lumière est utile pour réduire la possibilité d'images ambiguës. Il est dans ce cas préférable d'éviter d'examiner les nombreuses correspondances possibles. On peut positionner les sources le long d'alignements, et ces alignements peuvent être détectés dans les images. Avec des alignements de quatre sources chacun, on peut reconnaître et différencier les alignements en donnant à chacun des *rapports harmoniques* très différents. On peut aussi considérer des alignements de trois points; les rapports de distances pour trois points sont préservés de façon approximative en perspective, et si on choisit deux alignements avec des rapports très différents, la discrimination est généralement facile. Par exemple, nous avons construit un pointeur avec un alignement dont la source intermédiaire est à mi-distance des sources extrêmes, et un alignement dont la source intermédiaire est deux fois plus proche d'une source extrême que de l'autre (figure 8.8). Les choses sont compliquées par le fait qu'il arrive souvent que des alignements de taches claires qui ne correspondent à aucun alignement de sources du pointeur soient produits fortuitement dans une image (figure 8.8). Il faut donc trouver tous les alignements de trois points dans l'image, calculer les rapports de distance et choisir les meilleures correspondances. Ici encore, il est utile de vérifier les résultats en utilisant les poses précédentes

et la mesure d'erreur orthonormale.

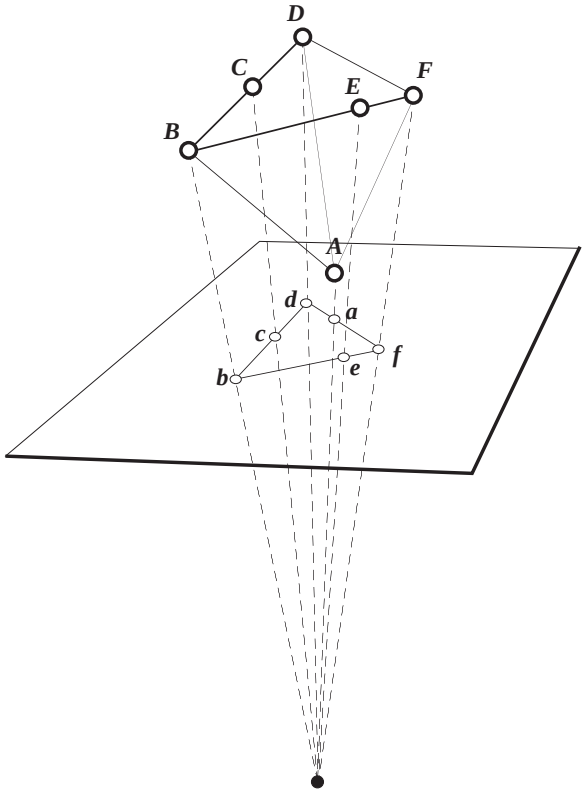

Figure 8.8: Objet comportant deux alignements coplanaires de sources de lumière, (B, C, D) et (B, E, F), et une source A en dehors du plan des alignements. La source A se projette ici dans l'alignement des images de D et F, créant un alignement (d, a, f) dans l'image qui ne correspond à aucun alignement dans l'objet, et compliquant pour cette pose la détermination des correspondances.

Une fois que quelques images ont été analysées et que quelques poses successives ont été obtenues, il peut être avantageux d'utiliser une méthode de poursuite, illustrée sur la figure 8.9, pour simplifier la mise en correspondance. A partir de ces poses précédentes, le système prédit approximativement la pose qui va être trouvée pour la nouvelle image, par simple extrapolation, ou par une technique plus sophistiquée, par exemple un filtre de Kalman. A partir de cette pose prédite, on peut obtenir les positions prédites des images des sources par projection perspective des sources pour cette pose. A partir des estimations des incertitudes de la prédiction de pose, on définit des carrés d'incertitude autour de chaque image de source prédite dans lesquels les centres des taches ont de grandes chances de se trouver. Considérant la liste des carrés d'incertitude et la liste des positions des taches

détectées dans la nouvelle image, on peut mettre en correspondance la source de lumière qui a servi à calculer un carré d'incertitude avec la tache claire qui se trouve dans ce carré. Les choses se compliquent lorsque deux carrés d'incertitude se superposent, comme sur la figure 8.9. Dans ce cas, il est possible d'avoir deux taches claires qui appartiennent chacune à deux carrés à la fois, et les deux sources de lumières qui ont servi à calculer ces deux carrés peuvent être mises en correspondance avec l'une ou l'autre des taches lumineuses. On pourrait alors résoudre cette ambiguïté en choisissant parmi ces correspondances celle qui minimise les mesures C_1 et C_2 définies ci–dessus. Mais on peut choisir l'approche plus pragmatique et plus rapide qui consiste simplement à ne pas chercher à utiliser les taches qui créent un problème, et à utiliser à leur place les positions des taches prédites, pour lesquelles on connaît bien sûr la correspondance. L'information relative aux positions des taches réelles qui sont ignorées est perdue, mais dans la prochaine image, il est probable que ces carrés d'incertitude ne se superposeront plus (tandis que d'autres carrés se superposeront peut-être), si bien que ces pertes d'information aléatoires tendent à se compenser [50].

On peut aussi utiliser une position prédite de tache à la place d'une position réelle lorsqu'un carré d'incertitude tombe entièrement ou partiellement en dehors de l'image, ou lorsqu'un carré d'incertitude ne contient aucune tache (souvent du fait que la source de lumière correspondante est cachée par un obstacle). Par conséquent, après les images initiales, cette méthode ne nécessite plus que toutes les sources de lumière soient visibles. On peut dire que dans ce cas le système "imagine" les sources de lumière quand il ne peut les voir [53].

La pose calculée se détériore si de plus en plus de sources de lumière deviennent invisibles et sont imaginées, jusqu'au moment où la pose calculée n'est plus assez précise pour poursuivre même les taches claires qui sont encore visibles. A ce moment, le système doit attendre que toutes les sources soient à nouveau visibles pour être capable de calculer quelques poses sans faire appel à une technique de poursuite, en appliquant une des techniques de correspondance décrites ci–dessus et pour pouvoir redémarrer une séquence de poursuite.

Nous avons programmé cette méthode de poursuite dans la première version de notre système. A l'époque (printemps 1992), notre matériel d'analyse d'image n'était pas très rapide, fournissant des positions de taches claires à 15 Hz environ (au lieu des 60 Hz

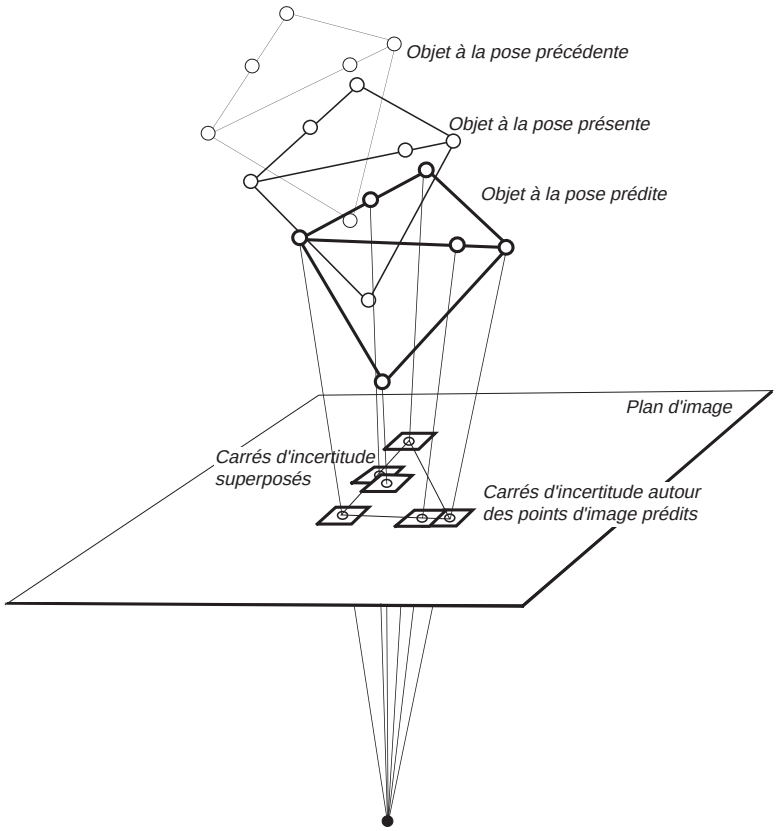

Figure 8.9: Méthode de correspondance par prédiction des positions des images des sources. Une image qui appartient au carré d'incertitude d'une source est appariée avec cette source. Lorsqu'il y a une ambiguïté entre deux sources et deux images, on utilise simplement les images prédites au lieu d'essayer de résoudre l'ambiguïté.

actuels). Cela nous obligeait à prendre de grands carrés d'incertitude pour tenter de suivre des accélérations rapides du pointeur, avec le résultat que les taches occupaient souvent plusieurs carrés. Nous avons observé un gain immédiat de robustesse dès que nous avons abandonné la poursuite dans l'image pour adopter les méthodes décrites au début de ce chapitre. Cela ne signifie pas que l'autre approche était mauvaise, mais simplement que notre premier programme n'était pas au point et demanderait davantage d'efforts. De tels programmes sont difficiles à déboguer car on ne peut observer les résultats sans altérer la fréquence de fonctionnement.

8.10 En résumé

Une géométrie de quatre sources de lumière dans laquelle les segments entre une source de référence et les trois autres sources sont égaux et perpendiculaires conduit à des calculs de pose très simples. La mise en correspondance entre sources et taches claires requiert quatre permutations pour la source de référence, puis six permutations des colonnes de la matrice de rotation. Le seul inconvénient de cette géométrie symétrique est que pour une image donnée, il existe deux poses symétriques possibles par rapport à un plan parallèle au plan d'image de la caméra. Il est en général facile de choisir la pose correcte par comparaison à une pose précédente. Mais l'ambiguïté de pose devient difficile à résoudre après la superposition des images de deux sources (autres que la source de référence), qui peut se produire quand l'axe du pointeur est perpendiculaire à l'axe optique de la caméra. Notre système évite ce problème en sélectionnant systématiquement la pose pour laquelle le pointeur est tourné vers la caméra, et avertit l'opérateur lorsque l'axe du pointeur devient perpendiculaire à l'axe de la caméra par un changement de couleur du curseur. Nous présentons d'autres méthodes de mise en correspondance, qui peuvent aussi être appliquées à des pointeurs comportant plus de quatre sources. Ces méthodes sont plus complexes mais permettent à l'opérateur de diriger le pointeur vers lui; les occlusions sont alors généralement plus fréquentes, mais le système peut continuer à tourner en utilisant des images prédites à la place des images des sources cachées.

Chapitre 9

Conclusions

La réalité virtuelle est un outil puissant d'interaction avec des données complexes, si on code les données de façon visuelle tridimensionnelle pour fournir à l'utilisateur la bande passante la plus large avec les zones cérébrales capables de raisonner sur ces données. Cet outil est appelé à remplacer dans un avenir proche l'univers plat des fenêtres, icônes et menus. Nous avons présenté un aperçu des promesses de ce domaine. L'interaction avec les données est effectuée surtout par l'intermédiaire de déplacements de la main et de la tête de l'opérateur, qui doivent être détectés et transmis à l'ordinateur par des systèmes de poursuite, ou "traqueurs". Nous avons passé en revue les techniques qui sont utilisées dans ces traqueurs, et nous avons comparé leurs paramètres de performance.

Par comparaison aux autres techniques, il est assez facile d'obtenir une réponse rapide avec un système optique, parce que les caméras vidéo courantes sont des capteurs qui fournissent des signaux à fréquence relativement élevée. Ces caméras sont en train de devenir bon marché. Nous avons construit un pointeur 3D qui utilise une seule caméra et fournit une solution peu coûteuse à réponse rapide. La construction d'un tel système nécessite la maîtrise de techniques à la fois dans les domaines de la vision par ordinateur, de l'interaction homme-machine, et du traitement du signal vidéo. Dans ces trois domaines, il a fallu développer des méthodes originales.

Les points caractéristiques du pointeur sont des sources de lumière infrarouge facilement détectées dans l'image vidéo. Nous avons développé un algorithme rapide, POSIT, pour

calculer la pose du pointeur à partir des images de ces sources. L'algorithme calcule d'abord une approximation de la pose qui suppose que les images ont été obtenues par un modèle de projection othographique à l'échelle. Cette étape ne requiert pour obtenir la matrice de rotation que deux produits de vecteurs par une matrice précalculée, la normalisation des vecteurs résultants, et un produit vectoriel, et pour la matrice de translation la multiplication d'un vecteur par la norme utilisée dans la normalisation mentionnée. L'étape suivante consiste à remplacer les images réelles par des images orthographiques à l'échelle, en utilisant la pose approchée de l'étape précédente. Ces deux étapes sont répétées jusqu'à ce que les points d'images calculés et discrétisés en pixels ne se déplacent plus.

Une géométrie de quatre sources de lumière dans laquelle les segments entre une source de référence et les trois autres sources sont égaux et perpendiculaires conduit à des calculs de pose particulièrement simples. La mise en correspondance entre sources et taches claires requiert l'examen de quatre permutations pour la source de référence, puis six permutations des colonnes de la matrice de rotation. Le seul inconvénient de cette géometrie symétrique est que pour une image donnée, il existe deux poses symétriques possibles par rapport à un plan parallèle au plan d'image de la caméra. Il est en général facile de choisir la pose correcte par comparaison à une pose précédente. Mais l'ambiguïté devient difficile à résoudre après la superposition des images de deux sources (autres que la source de référence), qui peut se produire quand l'axe du pointeur est perpendiculaire à l'axe optique de la caméra. Notre système évite ce problème en sélectionnant systématiquement la pose pour laquelle le pointeur est tourné vers la caméra, et avertit l'usager lorsque l'axe du pointeur devient perpendiculaire à l'axe de la caméra par un changement de couleur du curseur. Nous avons présenté aussi d'autres méthodes de mise en correspondance qui peuvent être appliquées à des pointeurs comportant plus de quatre sources. Ces méthodes sont plus complexes mais permettent à l'usager de diriger le pointeur vers lui; les occlusions sont alors généralement plus fréquentes, mais le système peut continuer à tourner en utilisant des images prédites à la place des images des sources cachées.

Nous avons présenté deux architectures pour la détection en temps réel de taches claires dans le signal provenant d'une caméra vidéo. Dans la première architecture, on met en mémoire tous les pixels, tout en marquant les lignes sans pixels clairs. Le microcontrôleur

profite du répit dans le signal vidéo à la fin de la transmission d'une trame pour lire les pixels clairs, et saute les lignes qui sont marquées vides. C'est la solution que nous avons choisie pour notre prototype. Dans la deuxième solution, on ne met en mémoire que les positions pour lesquelles les pixels passent du niveau bas au niveau haut et *vice-versa*. Le microcontrôleur profite du répit à la fin de la transmission de chaque ligne d'image pour lire les positions des rebords de tache claire pour cette ligne, et pour combiner cette information avec l'information acquise aux lignes précédentes, afin de trouver les centres des taches claires.

Comme pour le travail présenté ici, les efforts de développements futurs vont être dirigés dans trois directions: la vision par ordinateur, l'interaction homme-machine, et le traitement du signal vidéo:

Vision par ordinateur:

- Nous avons découvert dans la littérature sur les systèmes non linéaires et le chaos [22, 21] les outils qui vont nous permettre d'analyser les conditions de convergence de l'algorithme POSIT, tout au moins pour des configurations simples de points caractéristiques.

- Nous espérons aussi améliorer les performances de notre méthode de mise en correspondance automatique (chapitre 4). Nous avons besoin de visualiser des projections tridimensionnelles des espaces à quatre dimensions dans lesquelles les recherches de correspondances maximales sont effectuées, afin de voir les configurations de régions dans lesquelles l'algorithme perd du temps. Notre pointeur 3D sera utilisé pour naviguer dans cet univers de boîtes et de tranches qui s'intersectent.

- Avec une seule caméra, les erreurs dans le calcul de la composante de translation en profondeur et des composantes de rotations autour d'axes perpendiculaires à l'axe optique de la caméra sont trop importantes pour des applications de traqueurs optiques en chirurgie ou pour la numérisation précise de formes 3D. Nous cherchons à combiner les mesures de poses de plusieurs caméras pour minimiser ces erreurs.

Interaction homme-machine:

- Nous allons transporter du Mac à une machine Silicon Graphics Indigo un programme de peinture (2D) utilisant le pointeur 3D . Dans ce programme écrit par Yukio Fujii, l'opérateur utilise la translation latérale du pointeur pour déplacer son pinceau sur la page, la translation en profondeur pour changer la taille du pinceau, et les rotations pour changer la couleur de la peinture sur le pinceau, par une navigation dans l'espace des couleurs. Ainsi l'opérateur peut éviter les va-et-vient entre l'œuvre et la palette.

- Nous allons programmer la métaphore "véhicule volant" pour faciliter l'exploration de scènes 3D complexes.

- Nous allons construire un traqueur de tête en fixant une caméra miniature sur une paire de lunettes 3D à occlusions alternées des yeux, pour la machine Silicon Graphics sur laquelle notre pointeur 3D est maintenant installée; puis nous programmerons un système de vision stéréoscopique dépendant du point de vue de l'opérateur.

Traitement du signal vidéo:

Nous pensons pouvoir encore réduire le coût du système en utilisant une puce de logique programmable pour regrouper les pixels clairs.

Annexe A: Evaluations des erreurs angulaires

Dans nos évaluations de performance des méthodes de calculs de pose pour des configurations coplanaires et non coplanaires de points (chapitres 2 et 3), l'objet a son repère dans une orientation connue, et les algorithmes POS et POSIT calculent à partir de l'image de l'objet un repère qui est généralement dans une orientation différente. On veut trouver l'erreur angulaire du calcul, qui est la différence d'orientation entre les deux repères. On trouve d'abord l'axe de la rotation nécessaire pour aligner les deux repères. L'erreur angulaire est l'angle de rotation autour de cet axe nécessaire pour aligner le repère calculé avec le repère réel. L'axe de rotation et l'angle d'alignement peuvent être calculés en faisant appel aux quaternions, mais nous proposons une méthode qui semble plus directe. Considérant les deux vecteurs unitaires $\mathbf{i}$ et $\mathbf{i}'$ des deux repères, l'axe de rotation doit appartenir à un plan par rapport auquel les deux vecteurs unitaires sont symétriques. Ce plan est donc perpendiculaire au vecteur $\mathbf{i}'-\mathbf{i}$. De même, l'axe de rotation appartient au plan perpendiculaire aux vecteurs $\mathbf{j}'-\mathbf{j}$ et $\mathbf{k}'-\mathbf{k}$. L'axe de rotation doit donc avoir une direction $\mathbf{n}$ perpendiculaire à la fois aux trois vecteurs $\mathbf{i}'-\mathbf{i}$, $\mathbf{j}'-\mathbf{j}$ et $\mathbf{k}'-\mathbf{k}$. On peut donc faire une moyenne de produits vectoriels; on peut aussi considérer que les coordonnées de $\mathbf{n}$ satisfont le système d'équations linéaires composé de l'équation

$$(i'_x - i_x)n_x + (i'_y - i_y)n_y + (i'_z - i_z)n_z = 0$$

et de deux autres équations similaires pour $\mathbf{j}'-\mathbf{j}$ et $\mathbf{k}'-\mathbf{k}$; ce système peut être résolu par exemple par décomposition en valeurs singulières. On peut ensuite calculer l'angle de la rotation, qui est l'angle qui fait tourner le plan $(\mathbf{n},\mathbf{i})$ pour le rendre parallèle au plan $(\mathbf{n},\mathbf{i}')$, c'est-à-dire l'angle entre les produits vectoriels $\mathbf{n}\times\mathbf{i}$ et $\mathbf{n}\times\mathbf{i}'$. L'angle entre $(\mathbf{n},\mathbf{j})$ et $(\mathbf{n},\mathbf{j}')$, et l'angle entre $(\mathbf{n},\mathbf{k})$ et $(\mathbf{n},\mathbf{k}')$ sont probablement légèrement différents; on calcule donc la moyenne entre ces trois angles.

Annexe B: Variante pour POSIT

Nous résumons ici une variante de l'algorithme POSIT, pour des configurations non coplanaires de points (chapitre 2), qui ne requiert pas la détection de l'image de l'origine M_0 du repère de l'objet et calcule la matrice 4×4 de la pose.

Les équations à résoudre sont les équations 4.3 et 4.4. Les étapes de l'algorithme itératif de pose sont les suivantes:

1. ε_i= meilleure conjecture, où $\varepsilon_i = 0$ si aucune information de pose n'est disponible;

2. Début de boucle: Résoudre pour $\mathbf{I}$ et $\mathbf{J}$ dans les systèmes suivants

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x'_i, \mathbf{M}_0\mathbf{M}_i \cdot \mathbf{J} = y'_i$$

   avec

$$x'_i = x_i(1 + \varepsilon_i), \quad y'_i = y_i(1 + \varepsilon_i)$$

3. A partir de $\mathbf{I}$, trouver

$$\mathbf{R}_1 = (I_1, I_2, I_3),$$

$$T_z/f = 1/|\mathbf{R}_1|,$$

$$\mathbf{i} = (T_z/f)\mathbf{R}_1,$$

$$\mathbf{P}_1 = (T_z/f)\mathbf{I}$$

   Des opérations identiques fournissent $\mathbf{j}$ et $\mathbf{P}_2$ à partir de $\mathbf{J}$.

4. $\mathbf{k} = \mathbf{i} \times \mathbf{j}$, $\mathbf{P}_3 = (k_u, k_v, k_w, T_z)$, $\varepsilon_i = \mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_3/T_z - 1$

5. Si tous les termes ε_i sont assez proches des ε_i de la boucle précédente, aller à l'étape 6 de sortie; sinon, retourner à l'étape 2.

6. $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4$, $(\mathbf{P}_4 = (0, 0, 0, 1))$, sont les quatre rangées de la matrice de pose.

Nous examinons maintenant le calcul de $\mathbf{I}$ et $\mathbf{J}$ dans l'étape 2 de l'algorithme itératif. Par exemple, les équations pour $\mathbf{I}$ sont:

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x'_i$$

Les inconnues sont les quatre coordonnées (I_1, I_2, I_3, I_4), de $\mathbf{I}$, et on peut écrire une nouvelle équation pour chacun des points M_i pour lesquels on connaît le point d'image m_i et sa coordonnée en x, x_i. Cette équation a la forme $U_i I_1 + V_i I_2 + W_i I_3 + I_4 = x'_i$, où $(U_i, V_i, W_i, 1)$ sont les quatre coordonnées connues de M_i dans le repère de l'objet. Si on écrit de telles équations pour quelques points M_i, on obtient un système linéaire d'équations qui peut s'écrire sous forme matricielle $\mathbf{AI} = \mathbf{x}'$, où $\mathbf{A}$ est la matrice dont le i-ème vecteur ligne est $\mathbf{A}_i = (U_i, V_i, W_i, 1)$, et où $\mathbf{x}'$ est un vecteur colonne dont la i-ème coordonnée est égale à x'_i.

De manière équivalente, le vecteur $\mathbf{J}$ peut être calculé en résolvant le système linéaire $\mathbf{AJ} = \mathbf{V}_y$, où A est la même matrice, et $\mathbf{V}_y$ est un vecteur colonne dont la i-ème coordonnée est égale à y'_i.

Il y a quatre coordonnées inconnues pour $\mathbf{I}$ et $\mathbf{J}$, par conséquent $\mathbf{A}$ doit être au moins de rang 4 pour que le système fournisse des solutions uniques. Cette condition est satisfaite si la matrice $\mathbf{A}$ a au moins quatre rangées de coordonnées de points M_i, et si ces points ne sont pas coplanaires; par conséquent, la connaissance d'au moins quatre points caractéristiques non coplanaires et de leurs points d'image correspondants est nécessaire. L'opération de pseudo-inversion est appliquée à la matrice $\mathbf{A}$. La matrice pseudo-inverse, $\mathbf{B}$, est appelée "matrice d'objet". Du fait que la matrice $\mathbf{A}$ est définie par les coordonnées des points caractéristiques dans le repère de l'objet, la matrice d'objet $\mathbf{B}$ dépend uniquement de la géométrie des points caractéristiques et peut être précalculée.

Bibliographie

[1] Abidi, M. A., T. Chandra, "A New Efficient and Direct Solution for Pose Estimation Using Quadrangular Targets: Algorithm and Evaluation", Dept. of Electrical and Computer Engineering, The University of Tennessee, July 91, to be published in IEEE Trans. on Pattern Analysis and Machine Intelligence.

[2] Baird, H. S., *Model-Based Image Matching Using Location*, MIT Press, Cambridge, MA, 1985.

[3] Benson, B., *Television Engineering Handbook*, McGraw-Hill.

[4] Blood, E., "Three Dimensional Dgitizer with Electromagnetic Coupling", U.S. Patent no. 4,613,866, September 23, 1986.

[5] Breuel,T. M., "Model Based Recognition using Pruned Correspondence Search", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Maui, HI, pp. 257–262, 1991.

[6] Breuel, T. M., "Fast Recognition using Adaptive Subdivisions of Transformation Space", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Champaign, IL, pp. 445–451, 1992.

[7] Brooks, F. P., M. Ouh-Young, and J. Batter, "Project Grope –Haptic Displays for Scientific Visualization", Computer Graphics, 24 (4), pp. 175–185, 1990.

[8] Cass, T. A, "Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty", Computer Vision–ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp. 834–842, Springer-Verlag, 1992.

[9] Ciarcia, S., "Build a Gray-Scale Video Digitizer", Byte, May, June 1987.

[10] Davis, L. S., D. F. DeMenthon, T. Bestul, D. Harwood, H. V. Srinivasan, and S. Ziavras, "RAMBO: Vision and Planning on the Connection Machine", Image Understanding Workshop, pp. 631–639, May 1989.

[11] DeMenthon, D. F. and L. S. Davis, "New Exact and Approximate Solutions of the Three-Point Perspective Problem", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 14, no. 11, pp. 1100–1105, 1992.

[12] DeMenthon, D. F. and L. S. Davis, "Model-Based Object Pose in 25 Lines of Code", Proc. DARPA Image Understanding Workshop, San Diego, CA, pp. 753–761; also, Computer Vision–ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp. 335–343, Springer-Verlag, 1992.

[13] DeMenthon, D. F., "Computer Vision System for Position Monitoring in Three Dimensions using Non-Coplanar Light Sources Attached to a Monitored Object", U.S. Patent no. 5,227,985, June 13, 1993 (Application no. 07/747124, August 1991)

[14] DeMenthon, D. F., "Computer Vision System for Accurate Monitoring of Objet Pose", Patent Application no.08/098470, December 1992.

[15] DeMenthon, D. F., and Y. Fujii "Three Dimensional Pointing Device Monitored by Computer Vision", Patent Application no. 08/063489, May 1993.

[16] Dhome, M., M. Richetin, J. T. Lapreste, and G. Rives, "Determination of the Attitude of 3D Objects from a Single Perspective View", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 11, no. 12, pp. 1265–1278, December 1989.

[17] Egli, W. H., J. W. Miller, J. M. Setterholm, "Method and Apparatus for Determining Location and Orientation of Objets", U.S. Patent 4 672 562, June 1987.

[18] Faugeras, O. D., "A Few Steps toward Artificial 3 D Vision", INRIA Technical Report no. 790, February 1988.

[19] Fischler, M. A., and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Comm. of ACM, vol. 24, pp. 381-395, 1981.

[20] Fisher, S. S, "Living in a Virtual World", Byte, pp. 215–221, July 1990.

[21] Guckenheimer, J., end P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer-Verlag, 1983.

[22] Gulick, D., *Encounters with Chaos*, McGraw-Hill, 1992.

[23] Grimson, W. E. L and D. P. Huttenlocher , "On the Sensitivity of the Hough Transform for Object Recognition", Proc. International Conf. Computer Vision, Tarpon Springs, FL, 1988.

[24] Grimson, W. E. L, D. P. Huttenlocher, and D. W. Jacobs, "Affine Matching with Bounded Sensor Error: A Study of Geometric Hashing and Alignment", MIT AI Memo 1250, 1991.

[25] Grimson, W. E. L, D. P. Huttenlocher, and D. W. Jacobs, "Affine Matching with Bounded Sensor Error", Computer Vision–ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp. 291–306, Springer-Verlag, 1992.

[26] Haralick, R. M.,"Determining Camera Parameters from the Perspective Projection of a Rectangle", Pattern Recognition, vol. 22, no. 3, pp. 225–230, 1990.

[27] Haralick, R. M., C. Lee, K. Ottenberg, M. Nölle, "Analysis and Solutions of the Three Point Perspective Pose Estimation Problem", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Hawai, pp. 592–598, June 1991.

[28] Haralick, R. M.,"Performance Characterization in Computer Vision", University of Washington C.S. Technical Report, July 1991.

[29] Hayward, T., *Adventures in Virtual Reality*, Que Corp, 1993.

[30] Holt, R. J., A. N. Netravali, "Camera Calibration: Some New Results", CVGIP: Image Understanding, Vol. 54, no. 3, pp. 368-383, 1991.

[31] Horaud, R., B. Conio and O. Leboulleux, "An Analytical Solution for the Perspective-4-Point Problem", Computer Vision, Graphics, and Image Processing, vol. 47, pp. 33–44, 1989.

[32] Huttenlocher, D., S. Ullman, "Recognizing Solid Objects by Alignment", Proc. DARPA Image Understanding Workshop, pp. 1114-1122., 1988.

[33] Krieg, J. C., " A 4 Millisecond, Low Latency, 120 Hz, Electromagnetic Tracker for Virtual Reality Applications", Trade Literature, Polhemus, Inc., 1992.

[34] Lanier, J., dans la discussion "Virtual Environments and Interactivity: Windows of the Future", SIGGRAPH Panel Proceedings, 1989.

[35] Lowe, D. G., "Perceptual Organization and Visual Recognition", Kluwer Academic Publishers, 1985.

[36] Lowe, D. G., "Fitting Parameterized Three-Dimensional Models to Images", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 13, no. 5, pp. 441–450, May 1991.

[37] Lutton, E., "Reconnaissance du Point de Prise de Vue d'une Photographie à partir d'un Modèle de Scène", Thèse de Doctorat, Ecole Nationale des Télécommunications, Mai 1990.

[38] Masaaki, F., K. Mase, S. Yasuhito, "Finger-Pointer: A Glove Free Interface", Technical Description Sheet, NTT Human Interface Laboratories, Japan, 1991.

[39] McAvinney, "Telltale Gestures – 3D Applications need 3D input", Byte, pp. 237–240, July 1990.

[40] Meyer, K., H. L. Applewhite, anf F. A. Biocca, "A Survey of Position Trackers", Presence, vol. 1, no. 2, pp. 173–200, Spring 1992.

[41] Nisley, E., "ImageWise/PC – The Digitizing Continues", Circuit Cellar Ink, Nov.-Dec. 88, Jan.-Feb. 89, Feb.-March 89.

[42] Oberkampf, D., D. F. DeMenthon, and L. S. Davis, "Iterative Pose Estimation using Coplanar Feature Points", IEEE Conf. on Computer Vision and Pattern Recognition, New-York City, N.Y., June 15-18, 1993; full version: Center for Automation Research Technical Report CAR-TR-677, University of Maryland, July 1993.

[43] Ohya, J., D. F. DeMenthon, and L. S. Davis, "Recognizing Objects from Range Images and Finding their Position in Space", Third Int. Conf. of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Charleston, S.C., July 1990.

[44] Paley, W. B., "inScape Product Description", Digital Image Design Inc., Technical Literature, 170 Claremont Ave. Suite 6, NY, NY 10027, 1993.

[45] Peterson, I., "Looking-Glass Worlds", Science News, vol. 141, Jan. 4, 1992.

[46] Preparata, F. P., and M.I. Shamos, "Computational Geometry: An Introduction", Springer-Verlag, 1985.

[47] Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Veterling, Numerical Recipes in C, Cambridge University Press, Cambridge, UK, 1988.

[48] Robertson, G. G., S. K. Card, and J. D. Mackinlay, "Information Visualization using 3D Interactive Animation", Communications of the ACM, vol. 36, no. 4, pp. 57–71, April 1993.

[49] Shneiderman, B., *Designing the User Interface, Strategies for Effective Human–Computer Interfaces*, Second Edition, Addison Wesley, 1992.

[50] Silvén, O., "Estimating the Pose and Motion of a Known Object for Real-Time Robotic Tracking", Center for Automation Research Technical Report CAR-TR-503, University of Maryland, April 1990.

[51] Stephenson, N., *Snow Crash*, Bantam Books, 1992.

[52] Sutherland, I. E., "A Head Mounted Three Dimensional Display", Proc. Fall Joint Computer Conference, vol. 33, pp. 775–764, 1968.

[53] Tomasi, C., "Shape and Motion from Image Streams: A Factorization Method", Technical Report CMU-CS-91-172, Carnegie Mellon University, September 1991.

[54] Tsai, R.Y.,"A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," IEEE J. Robotics and Automation, vol. 3, pp. 323–344, 1987.

[55] Ullman, S., and R. Basri, "Recognition by Linear Combinations of Models", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 13, no. 1, pp. 992-1006, 1991.

[56] Waldren, J. D., "Method and Apparatus for the Perception of Computer-Generated Imagery", U.S. Patent no. 4,884,219, November 28, 1989.

[57] Wang, J, V. Chi and H. Fuchs, "A Real-time Optical 3D Tracker for Head Mounted Display Systems", Computer Graphics: Symposium on Interactive 3D Graphics, 24 (2), pp. 205–215, March 1990.

[58] Ware, C., and S. Osborne, "Exploration and Virtual Camera Control in Virtual Three Dimensional Environments", Proc. SIGGRAPH, pp. 175–183, 1990.

[59] Weber, J., "Seeing is Believing", Byte, pp. 121–128, April 1993.

[60] Wohlers, T. T., "3D Digitizers", Computer Graphics World, pp. 73–106, July 1992.

[61] Wolf, P.R., *Elements of Photogrammetry*, McGraw Hill, New-York, 1974.

[62] Wolfe, W. J., D. Mathis, C. W. Sklair, M. Magee, "The Perspective View of Three Points", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 13, no. 1, pp. 66–73, 1991.

[63] Yuan, J.S.C., "A General Photogrammetric Method for Determining Object Position and Orientation", IEEE Trans. on Robotics and Automation, vol. 5, no. 2, pp. 129–142, 1989.

[64] Zeevi, Y., and O. Hileerath, "Single Camera Three Dimensional Head Position Sensing System", U.S. Patent 4,956,794, September 1990.

[65] Zimmerman, T. G., and J. Z. Lanier, "Computer Data Entry and Manipulation Apparatus and Method", U.S. Patent no. 4,988,981, January 29, 1991.

[66] "Hand Master Controls "Smart" Machines", NASA Tech Briefs, vol. 13, no. 10, October 1989.

[67] Internet, sci.virtual-reality newsgroup, July 1993.

[68] "Substituting a Finger for a Mouse", New-York Times article, December 7, 1992, describing U.S. Patent 5,168,531 to C. Sigal.

[69] VLSI Vision, Ltd, Trade literature, Aviation House, 31 Pinkhill, Edinburgh EH12 8BD, UK, FAX 44 31-539 7140.

Liste des Figures