# MISE EN PLACE D'OUTILS DE TÉLÉOPÉRATION ROBOTIQUE

**EL JALAOUI Abdellah**

30 juin 2003

# RÉSUMÉ

**Résumé** – *Le travail présenté ici a été réalisé au cours d'un stage ingénieur et DEA effectué au sein du laboratoire* ***INRIA Rhône-Alpes*** *(**I**nstitut **N**ational de **R**echerche en **I**nformatique et en **A**utomatique). Il consiste en la mise en place d'un système de téléopération robotique. Souvent utilisé pour opérer dans les milieux hostiles à l'homme (centrale nucléaire, sous-marins), la téléopération consiste à faire exécuter à un robot les mouvements d'un opérateur situé dans un site protégé.*

*Le système de téléopération robotique mis en oeuvre pendant le stage utilise trois différents outils de capture de mouvement. Ces outils fournissent une suite de données traduisant la position au cours du temps du bras maître. Un filtrage spatial et temporel avant l'envoi des données au dispositif robotique permet d'éviter de surcharger le réseau. De ces données de position exprimées dans le repère d'acquisition, on extrait un mouvement qui sera reproduit sur le dispositif robotique.*

**Abstract** – *The work presented here has been carried out during a training course of engineer and DEA achieved within the laboratory* ***INRIA Rhône-Alpes*** *(**I**nstitut **N**ational de **R**echerche en **I**nformatique et en **A**utomatique). It consists on the realisation of a teleoperated robotic system. Often used to operate in hostile environnements (nuclear thermal power station, submarines), teleoperation consists in making movements of a user located in a protected site, to a robotic system.*

*Teleoperated system achieved during the training course use three motion capture divices. These tools provide data relating to the position of master arm. Data are filtered before being sent in order to avoid overloading the network. From these data expressed in the capture reference mark, we extract a movement which is to be reproduced on a robotic system.*

# TABLE DES MATIÈRES

# TABLE DES FIGURES

# REMERCIEMENTS

*Je tiens à exprimer toute ma gratitude à mes encadrants, Roger Pissard-Gibolet et Soraya Arias pour m'avoir guidé tout au long du stage dans mes décisions et avoir mis a ma disposition leurs connaissances dans le domaine de la robotique et surtout de l'informatique.*

*Mes remerciements s'adressent également aux autres personnes du laboratoire. En commençant par Sebastien Jarde pour son aide et ses conseils sur l'environnement UNIX et LINUX. Puis Jean-François Cuniberto pour avoir mis a ma diposition du matériel informatique et avoir maintenu l'environnement de travail.*

*Je désire également remercier mon tuteur ISIM, Etienne Dombre pour sa confiance et ses conseils et son aide précieuse en robotique.*

# INTRODUCTION

Le démantèlement d'un réacteur nucléaire ou encore la réparation d'un engin dans l'espace nécessitent l'intervention d'un opérateur humain, tout en présentant des conditions qui lui sont hostiles. La téléopération a remédié à ce problème en mettant en place dans le site dangereux des dispositifs robotiques capables de reproduire les mouvements d'un opérateur distant.

La téléopération pour ce type d'application consiste en général en un robot manipulateur, situé dans le site à risque, (site esclave), reproduisant le mouvement d'un joystick, actionné par un opérateur depuis un site protégé (site maître). On trouve souvent dans ce schéma l'utilisation de systèmes à retour d'effort.

Dans le cadre de mon stage, il m'a été demandé d'explorer la téléopération de robots en mettant en oeuvre des systèmes utilisés en réalité virtuelle, basés sur la capture de mouvement. L'utilisation de tels dispositifs est complémentaire à l'utilisation de systèmes à retour d'effort et devrait permettre d'enrichir la téléopération.

Le document se divise en quatre chapitres : le premier est un état de l'art sur la téléopération orienté sur les dipositifs utilisés. Dans le deuxième chapitre, je présente les spécifications du système à réaliser. Ensuite je détaille la conception et la réalisation avant de présenter les résultats expérimentaux.

# 1

# ÉTAT DE L'ART

## 1.1 Introduction

Les différentes techniques de téléopération, qui permettent à un opérateur humain d'accomplir une tâche à distance, sont construites selon le même principe de base à savoir : un système robotique d'intervention, qui exécute réellement la tâche, commandé à partir d'une station de contrôle par l'intermédiaire d'un canal de télécommunication. Leur mise en oeuvre répond souvent à un besoin sécuritaire tel la manipulation d'objets dans des environnements hostiles, milieux à fort taux de radioactivité, espace, milieux sous-marins, etc.., ou encore à un besoin de précision comme pour les applications relatives au milieu médical.

Les méthodes mises en oeuvre pour réaliser une téléopération sont fonction du type de tâche à effectuer ainsi que des contraintes de temps, principalement imposées par le médium de transmission.

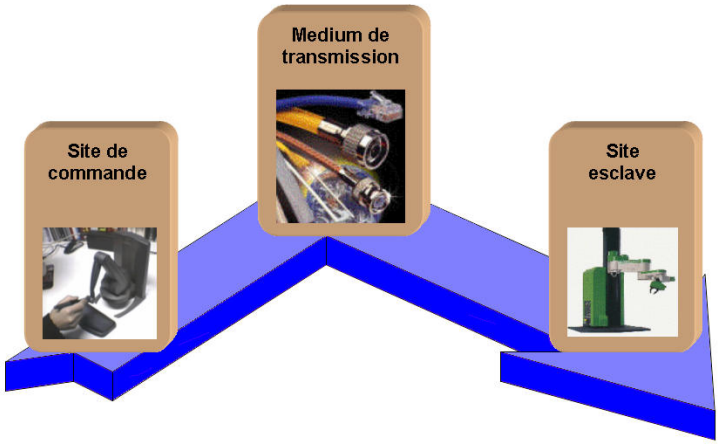



FIG. 1.1 – Téléopération

## 1.2 Techniques actuelles

Les dispositifs de téléopération développés pour des domaines aussi variés que l'éducation, l'industrie ou encore la médecine répondent à des contraintes différentes et n'atteignent pas les mêmes performances. Toutefois, ces systèmes sont toujours construits sur une base commune optimisée selon un sous-ensemble de critères particuliers (temps, précision, sécurité, accessibilité,...) imposé par le domaine d'application. Néanmoins, l'importance du délai de transmission entre les sites marque une distinction architecturale notable parmi les méthodes mises en oeuvre.

### 1.2.1 Faibles contraintes de temps

Dans le cas d'applications développées à des fins éducatives (voir [16], [14] et [9]), elles ne nécessitent pas une communication en temps réel entre le site de commande et le site distant, il s'agit le plus souvent d'envoyer un ordre de déplacement puis d'observer après coup le résultat. L'intérêt est surtout porté sur les différents états du système esclave et moins sur son évolution au cours du temps. Il n'y a pas un unique site de commande mais plusieurs, souvent très simples : un ordinateur connecté à Internet.

Etant donné le besoin de ce type de téléopération : des sites de commandes multiples localisés sur différents endroits du monde, dans tous les cas c'est l'Internet qui est utilisé comme medium de transmission. Les études concernant ce sujet s'attachent à améliorer les logiciels gérant les interfaces avec Internet, notamment au niveau de leur portabilité.

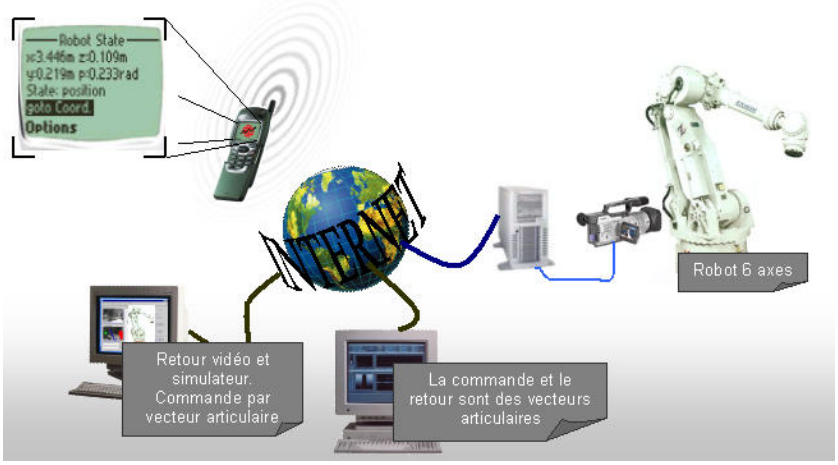
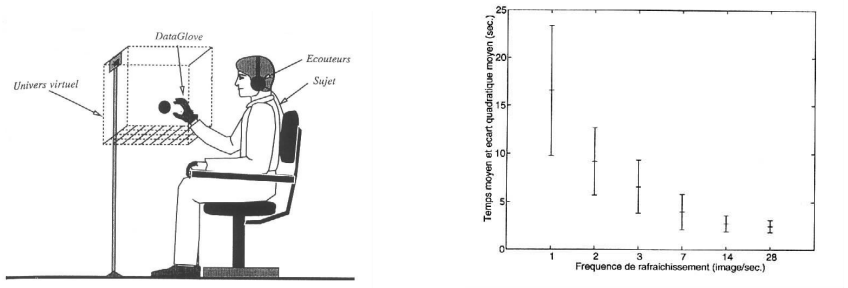



FIG. 1.2 – Téléopération via Internet

Les architectures les plus souvent rencontrées comportent sur leur site esclave un dispositif robotique, dans la majorité des cas un robot manipulateur six

axes, relié à un serveur web. Les données de commandes sont issues d'un poste connecté à Internet. L'opérateur saisit sur son PC le mouvement à faire faire au robot et le tout est envoyé au site distant. Celui-ci lui envoie à son tour sa configuration sous forme de valeurs d'articulations, par exemple, ou encore une image vidéo du dispositif.

L'utilisation de la vidéo, augmente la performance de l'utilisateur mais requièrt beaucoup de bande passante. Elle est donc mal adaptée pour l'émission sur Internet. Il faut souvent jouer avec la qualité de l'image et son taux de rafraîchissement pour répondre aux contraintes. Mais d'après une expérience rapportée dans [6] : en deçà de 14 images par seconde, les performances de l'opérateur se dégradent de façon exponentielle, voir figure 1.3(b). Cette expérience, illustrée à la figure 1.3(a), consiste à mesurer le temps mis par un opérateur à saisir une balle qui se déplace dans une scène virtuelle selon que le taux de rafraîchissement de l'image soit de 1,2,3,7,14 ou 28 images par seconde.




(a) Dispositif (b) résultats

FIG. 1.3 – Dextérité et taux de rafraîchissement

Dans l'implémentation des logiciels de téléopération, on remarque une prédominance du langage JAVA pour ce type d'application. En effet il est très bien adapté pour l'interfaçage avec l'Internet, mais aussi d'après [9], il est simple, orienté objet, largement distribué, robuste, sécurisé, portable et multitâche. Dans le cas de téléopérations plus complexes, où interviennent par exemple du traitement d'image ou l'utilisation de langages naturels pour la commande, on se contente de reprendre les applications dans leur langage de base et de les interconnecter, par exemple [14] utilise le standard Corba à cette fin.

Dans [14], R.Marin, P.J. Sanz et A.P. del Pobil, ont ajouté la possibilité d'utiliser des commandes en langage évolué comme "prend le petit stylo". Dans [16], le retour d'infomation vers l'utilisateur se fait par le moyen de la vidéo temps réel à 150 Kbps. Une équipe a réalisé en complément d'un retour vidéo, un simulateur de robot manipulateur qui permet de visualiser l'état du robot selon les données reçues et selon les ordres envoyés [9]. Cela permet, entre autre, de compenser la mauvaise qualité et le retard dans l'utilisation de la vidéo sur

Internet. Enfin dans [13] et [12], les auteurs présentent des outils utilisant la technologie WAP pour réaliser la commande d'un dispositif robotique depuis un téléphone cellulaire.

### 1.2.2 Fortes contraintes de temps

Les téléopérations, utilisées en industrie, se doivent de respecter certaines règles importantes de ce milieu comme la sécurité et la précision dans les différentes tâches pour lesquelles elles sont employées. Pour un meilleur contrôle, il devient nécessaire à l'opérateur humain de se sentir le plus possible immergé, à tout instant, dans l'environnement distant pour répondre au mieux à ses contraintes. Pour cela, on s'attache à ce que les temps de transfert de données entre le site maître et le site esclave n'affectent pas la transparence du dispositif. Pour la commande, on essaye de restituer à l'utilisateur le maximum d'informations, comme le retour d'efforts, qui lui permet d'évoluer aisément et répondre de manière plus adéquate aux contraintes subies par le manipulateur esclave.

Les architectures retenues pour ce type de téléopération confèrent au medium de transmission une importance tout aussi grande que pour le site de commande ou le site esclave. Le site de commande comporte dans la plupart des cas un joystick avec retour d'effort, un simulateur et un retour visuel. Le site distant est un manipulateur six axes muni d'une unité de traitement. La liaison maître esclave dépend du milieu d'application (sous-marin, nucléaire, spatiale) et est choisie d'abord selon le délai de transmission qu'engendre leur utilisation.

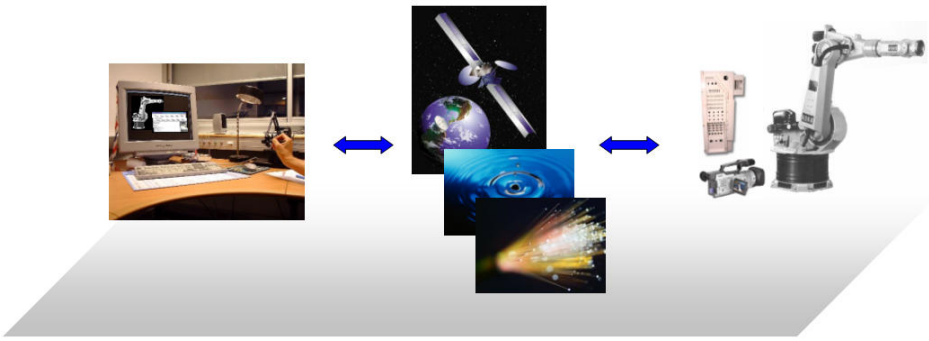


FIG. 1.4 – Importance du medium dans les téléopérations pour l'industrie (Ondes électromagnétiques pour l'espace, onde sonore dans l'eau, fibre optique)

Les études menées sur ce sujet, concernent principalement l'amélioration de l'architecture de commande en tenant compte du délai de transmission et notamment en vue de permettre l'intégration d'un retour d'effort. Le retour d'effort est une source d'information très riche pour l'utilisateur. En effet, elle augmente de façon significative sa dextérité et son aptitude à évaluer l'ampli-

tude des interactions entre le manipulateur esclave et son environnement. Dans [6], est décrite une expérience visant à comparer la dextérité des sujets en ajoutant à leur vision directe de la scène, une assistance visuelle, auditive, haptique ou sans aucune assistance. Les sujets, munis d'un gant sensitif, sont invités à saisir dans une scène virtuelle une balle, de l'écraser jusqu'a ce qu'elle atteigne 10% de son rayon et de la reposer dans une position précise sans la lâcher. L'assistance visuelle consiste en une rangée de 20 LEDs dont le nombre allumé est proportionnel à la déformation. Le retour auditif est un signal sonore dont la fréquence est proportionnelle à la déformation.

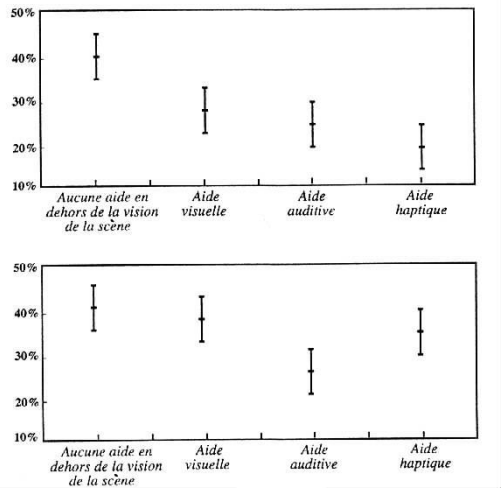



FIG. 1.5 – Niveau moyen et écart type de la déformation atteinte en fonction des diverses assistances perceptives. En haut avec une balle dure, en bas, avec une balle molle.

G. Burdea et P. Coiffet [6] concluent sur les résultats de cette expérience que :

> ...l'aide auditive et l'aide haptique améliorent le taux de succès [...] mais aucune de ces aides n'est suffisante pour un contrôle de grande qualité et que le retour d'effort doit certainement être amélioré puisqu'on obtient pas les mêmes résultats sur l'efficacité des aides avec la balle dur et avec la balle molle (l'aide haptique et l'aide auditive sont inversées).

L'amélioration de l'efficacité du retour d'effort passe par l'aptitude de l'architecture à rendre la chaîne quasiment transparente pour l'opérateur. La qualité de cette transparence repose sur la stabilité de l'architecture de commande du système esclave dont le principal obstacle est le délai de communication. Si ce dernier est supérieur à environ 150 à 200 ms, on ne peut pas faire de retour d'effort. S'il est aléatoire, comme sur un réseau par exemple, se pose des problèmes

d'instabilité. Ainsi, plusieurs architectures de contrôle ont été développées pour permettre d'assurer la stabilité du système [7] et [3].

Le système est modélisé comme sur la figure 1.7, il est représenté par un quadripôle électrique dont les tensions aux bornes et les courants représentent respectivement des forces et des vitesses. Ainsi, rendre le dispositif de téléopération transparent vis-à-vis de l'opérateur, c'est rendre le quadripôle modéle transparent électriquement. Du fait que le système est multivariable, il est souvent très difficile de trouver une condition nécessaire et suffisante de stabilité, on essaye plutôt de trouver une architecture passive, la passivité étant une condition suffisante de stabilité [3].


FIG. 1.6 – Quadripôle modèle

F. Geffard, C. Andriot, A. Micaelli et G. Morel [7] proposent deux architectures de commande d'un dispositif de téléopération à retour de force basé sur un manipulateur RX90. Ces architectures permettent de passiver et donc de rendre stable le système selon que le capteur d'effort du bras de robot est attaché à son poignet ou à sa base [7]. D'autres problèmes, propres aux domaines d'applications, ont été étudié, comme la conception par une équipe coréenne [5] d'une architecture de commande visant à éliminer le problèmes des bruits parasites dans une téléopération sous-marine. En effet, en milieu sous-marin, le courant de l'eau impose sur le manipulateur des forces parasites dont l'amplitude varie au cours du temps [5].

Le manipulateur esclave en chirurgie à la différence des autres domaines d'application, est en contact avec des surfaces déformables. Pour détecter le changement spatiale de compliance des tissus, comme à l'abord d'une tumeur par exemple, la chaîne de téléopération doit assurer assez de transparence et pouvoir restituer le contraste faible des forces appliquées par le milieu au manipulateur [11].

La téléopération est maintenant effective dans plusieurs domaines industriels et notamment le domaine nucléaire pour lequel elle est devenue incontournable. Pour démonter des filtres et des gaines de ventilations dans une salle où le taux de radioactivité ne permet pas l'intervention humaine, la Comex a développé une machine mobile, munie d'un bras téléopéré. Le bras est constitué de 5 axes linéaires et rotatifs et peut s'équiper sans intervention humaine de divers outils de coupe. Ce bras peut être commandé avec un bras maître ou en mode cartésien.

Pour l'exploration des grands fonds marins, l'Ifremer a développé un système téléopéré, Victor 6000, (voir figure 1.8(a)) qui est un sous-marin muni de deux bras articulés (voir figure 1.8(b)). Un câble en fibre optique le relie à la surface et permet ainsi sa commande et sa supervision.

Les logiciels destinés aux téléopérations pour application industrielle, sont

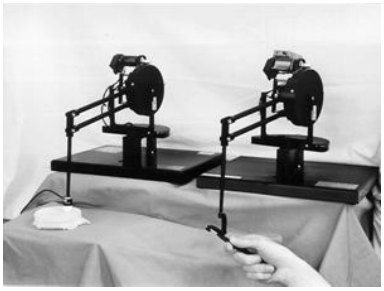
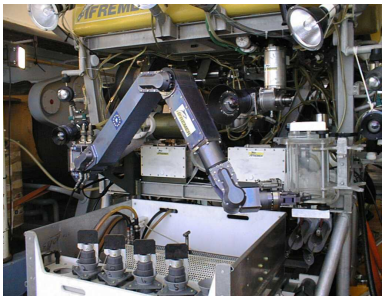


FIG. 1.7 – Téléopération haute fidélité


(a) Victor 6000



(b) Bras manipulateurs

FIG. 1.8 – Système téléopéré pour l'exploration des grand fonds marins (IFREMER)

généralement développés et optimisés pour une application donnée. Toutefois, la société Trasys a développé un logiciel—DREAMS— de commande et de supervision standard. Conçu initialement pour la robotique spatiale, il présente une interopérabilité lui permettant d'être interfacé avec la plupart des dispositifs de téléopération.

La fonction principale du site de commande dans une téléopération est de capturer le mouvement humain puis de le transmettre au site esclave. La plupart des dispositifs de téléopération actuel, utilisent pour récupérer le mouvement de l'utilisateur un joystick plus ou moins sophistiqué. Des systèmes de capture de mouvements développés pour d'autres domaines comme le cinéma ou la réalité virtuelle ont atteint un niveau de performance et de fiabilité satisfaisant. Ils peuvent tout-à-fait être utilisés dans le cadre d'une téléopération et ainsi améliorer la qualité de l'opération.

## 1.3 Systèmes de capture de mouvements

Les mouvements de l'opérateur humain sont enregistrés puis diffusés vers l'organe effecteur esclave distant qui doit alors les reproduire. Plusieurs systèmes ont été développés afin d'effectuer l'acquisition de mouvements. On peut les classer en trois grandes catégories construites en fonction de l'emplacement des capteurs vis-à-vis de l'opérateur et de son environnement [2]et[8] :

**1- les systèmes extérieur-entrants :** (Outside-in systems) ils utilisent des capteurs externes pour récupérer des signaux émis par des sources placées sur le corps de l'opérateur qui effectue le mouvement. Généralement, ces systèmes emploient des caméras comme capteurs et des marqueurs réfléchissants comme sources.

**2- les systèmes intérieur-sortants :** (Inside-out systems) ils utilisent des capteurs placés sur le corps de l'individu, qui reçoivent les informations d'émetteurs externes. Dans ce type de systèmes, la personne en mouvement porte sur lui des récepteurs électromagnétiques et se meut dans un environnement dans lequel on a recréé un champ électromagnétique.

**3- les systèmes intérieur-entrants :** (Inside-in systems) ils se basent sur les sources et les récepteurs placés sur le corps de l'acteur, par exemple, les combinaisons électromécaniques dans lesquelles les capteurs sont des potentiomètres ou des goniomètres et l'émetteur est constitué par les articulations du corps humain.

### 1.3.1 Capteurs de nouvements mécanique

Les dispositifs mécanique d'acquisition de mouvements sont généralement constitués d'une combinaison munie d'un exosquelette, c'est-à-dire un ensemble de tiges articulées (voir figure 1.9). Les mouvements du corps entraînent ceux des tiges articulées ; des capteurs d'angles, tels que des potentiomètres placés aux articulations de l'exosquelette permettent de recueillir des informations sur les mouvements du corps.

Certains autres dispositifs attachés à une partie spécifique du corps, la main par exemple, sont constitués d'un revêtement (gant) contenant des capteurs d'efforts. Ces capteurs envoient des informations sur les forces exercées par l'utilisateur bougeant ses doigts à un ordinateur, qui en extrait les mouvements en utilisant une modélisation de la main [1].

Du fait de la disposition même des capteurs, les informations issues de ces systèmes ne permettent de calculer que la disposition relative des différentes parties du corps les unes par rapport aux autres. Ce

FIG. 1.9 – Gypsy3.1

dispositif décrit alors la déformation du corps mais
pas sa position par rapport à l'environnement. On
a souvent recours à des capteurs électromagnétiques pour connaître la translation globale du système, mais cela ajoute une contrainte importante liée au fonctionnement de ce type de capteur, à savoir la nécessité d'écarter tout objet métallique de la scène d'acquisition.

Enfin, les modèles de base utilisés pour concevoir ces dispositif supposent que toutes les articulations du corps sont du type pivot [10, p52].

### 1.3.2 Capteurs de mouvements par ultra-sons

Ces systèmes sont constitués d'émetteurs ultrasonores et de récepteurs du même type. Les uns sont fixés sur l'organe physique dont on veut connaître la position dans l'espace, les autres sont fixés à la scène. Chaque émetteur est constitué d'un triplet de haut-parleurs à ultrasons disposés en triangle et de manière analogue, chaque récepteur comporte trois microphones montés sur un bâti rigide triangulaire. En activant les haut-parleurs successivement, on note le temps mis par le son pour arriver à chaque microphone et connaissant la vitesse du son dans l'air, on en déduit les distances de chaque microphone à chaque haut-parleur. Ces mesures permettent de connaître la position et l'orientation du plan contenant les trois émetteurs [10, p54] et [6, p26–28].

FIG. 1.10 – Souris à ultra-sons Logitech

### 1.3.3 Capteurs de mouvements magnétiques

Ces systèmes se basent sur le principe suivant : un repère orthogonal formé par trois antennes fixées à la scène génèrent un champ électromagnétique modulé à basse fréquence et sert de source à d'autres triplets d'antennes situés dans des récepteurs. Les récepteurs sont généralement placés sur les articulations d'un sujet humain. Le système analyse les données des capteurs et fournit en temps réel leurs positions ainsi que leurs orientations. La contrainte majeure imposée par la nature des signaux utilisés est que la scène où se fait l'acquisition doit être exempte d'objets métalliques [6, p19–26].

### 1.3.4 Capteurs de mouvements vidéo

On distingue principalement trois types de dispositifs à base de caméra :

**1- Sans marqueurs :** Ces dispositifs ne placent aucun marqueur sur le sujet mais se contentent de traiter l'image issue de la scène pour en déduire les mouvements de ce dernier.

**2- Marqueurs actifs :** Des marqueurs actifs tels des diodes sont placés à des points clefs du corps de l'individu. Ils peuvent être activés sequentiellement, ce qui évite de les confondre.

**3- Marqueurs passifs :** Le plus souvent des sphères recouvertes d'un matériau réfléchissant, les marqueurs passifs peuvent être utilisés en grand nombre et présentent un faible encombrement mais requièrent souvent beaucoup plus de traitement au niveau de l'unité de calcul.

#### 1.3.4.1 Sans marqueur

Le système informatique reçoit à un instant donné les images de l'individu prises selon différents angles de vue par les caméras. Il essaye d'identifier chaque partie du corps du sujet filmé à la partie correspondante sur un modèle géométrique présent dans une base de données. Ces dispositifs bien que non contraignants sont encore délicats à mettre en place et du fait des lourds algorithmes de traitement d'image qu'ils emploient, ils ne peuvent fournir de données en temps réel [10, p52–54]. La société InteraMotion a conçu une plateforme d'analyse de mouvement basée sur des cameras digitales, qui permet de reconnaître en temps réel des mouvements simple de combat.

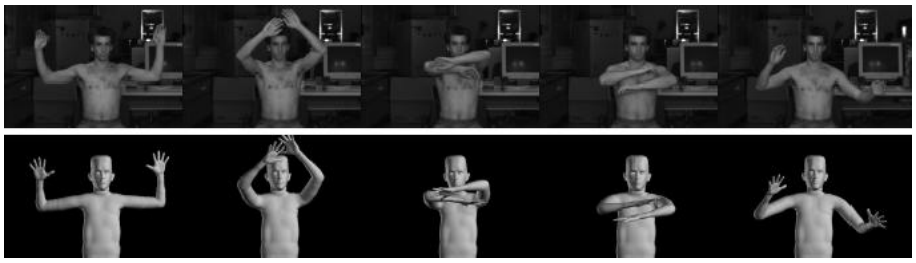


FIG. 1.11 – Identification du corps humain

#### 1.3.4.2 Marqueurs actifs

Les marqueurs actifs sont le plus souvent des diodes infrarouges placées aux articulations du sujet et activées séquentiellement. Un ensemble de plusieurs caméras capte les signaux émis par chaque diode. Le calculateur confronte les positions 2D des diodes, fournies par chaque caméra pour en déduire la position 3D. Les caméras sont souvent munies d'un filtre infrarouge pour ne voir que les diodes et alléger ainsi la phase de traitement d'image.

#### 1.3.4.3 Marqueurs passifs

Les marqueurs passifs sont généralement composés d'une sphère de 1 à 5 cm de diamètre recouverte d'un matériau réfléchissant. Les caméras disposées sur la scène envoient les images à un système informatique qui doit distinguer chaque marqueur, puis extraire des images leur position respective, en tenant compte du fait que certains marqueurs peuvent être occultés.Les marqueurs passifs présentent aussi le défaut de ne pas être utilisables en extérieur à cause des interférences avec les radiations solaires. On ne peut, en général, pas obtenir de données en temps réel à cause de l'important traitement d'image à effectuer. Cependant, la société Vicon a développé un dispositif permettant de fournir des données en temps réel [1] et [15].

| Type de système | Matériel | | Caractéristiques typiques | Avantages Inconvénients |
|---|---|---|---|---|
| extérieur-entrant | Vidéo | Sans marqueurs | Pas d'acquisition en temps réel. | Aucune contrainte sur l'individu et la scène, très faible coût matériel. Données très complexes à analyser, faible précision. |
| | | marqueurs actifs | **Précision** : 0.1 mm. **Bande passante** : 600 Hz. **Portée** : 10 m **coût** : 100 000 $-250 000 $ | Traitements informatiques faibles, bonne précision. Encombrement des marqueurs, pas de mesures en extérieure. |
| | | marqueurs passifs | **Précision** : 0.1 mm **Bande passante** : 200 Hz. **Portée** : 30 m **coût** : 100 000 $-250 000 $ | Bonne précision, possibilité d'utiliser beaucoup de marqueurs, pas de gêne de l'opérateur, grand champ d'acquisition. Lourd traitement pour distinguer les marqueurs, capture impossible si marqueurs occulte trop longtemps. |
| intérieur-sortant | Magnétique | | **Précision** : 0.2 mm. **Bande passante** : 60 Hz. **Portée** : 3 m **coût** : 5 000 $-150 000 $ | Fournit des données en temps réel, positions et orientations disponibles sans traitements, bon marché. Contrainte sur la zone d'acquisition : pas d'objets métalliques, zone d'acquisition réduite. |
| | Ultrasonore | | **Précision** : 5 mm. **Bande passante** : 60 Hz. **Portée** : 1 m **coût** : 3 000 $ | Systèmes très bon marché. Nécessite une ligne directe entre l'émetteur et le récepteur. Sensibles qu bruit de fond et variations de température. |
| intérieur-entrant | Electromécanique | | **Précision** : 0.5°. **Bande passante** : 70 Hz. **Portée** : 100 m **coût** : 30 000 $ | Scène de capture très étendue, dispositifs bon marché, aucune contrainte sur la zone de capture. Basse fréquence d'acquisition, combinaison contraignante pour l'individu, mouvements limité, pas de mesure de la translation globale. |

FIG. 1.12 – Tableau recapitulatif

## 1.4 Les retours d'informations

Toutes les techniques récentes de capture de mouvement—mis à part les dispositifs vidéo sans marqueur—peuvent fournir la posture d'un individu en temps réel et sont donc candidates pour la mise-en-oeuvre d'un système de téléopération. Pour que la téléopération soit effective, il reste à transférer ces informations

de position au site esclave distant. Les propriétés du médium de transmission jouent alors un rôle dans la faisabilité et la stabilité de la téléopération. Si le temps de transmission est trop long, il est impossible à l'opérateur de visualiser ce que le dispositif robotique esclave effectue. Dans le cas d'une téléopération utilisant le retour d'effort (permet à l'opérateur de ressentir l'environnement de travail du site esclave), il faut adapter l'architecture de la commande du système pour prendre en compte ce temps de retard et rendre le système stable. Parfois, lorsque ce dernier s'avère trop important (plus de 150 ou 200 ms), le retour d'effort devient inutilisable.

Les systèmes à retour d'effort et tactiles permettent d'obtenir des informations concernant la topographie. Ces dispositifs permettent d'augmenter la dextérité de l'opérateur humain, notamment pour le milieu médical : l'entraînement des chirurgiens se fait sur des corps virtuels, sans retour visuel mais avec retour d'effort.

Dans la vie quotidienne, les retours d'effort et tactiles sont une source importante d'informations sur la topographie du monde environnant. On augmente de manière significative la dextérité de l'opérateur humain en utilisant un dispositif muni d'un retour d'effort.

### 1.4.1 Retour tactile

Les retours tactiles sont à distinguer des retours d'effort par la nature des informations qu'ils sont capables de fournir. Les capteurs tactiles donnent une information sur les caractéristiques de surface d'un objet : rugosité, température, glissement. La main, organe très sensible au toucher, est l'interface privilégié pour recevoir un retour tactile. Elle possède une résolution spatiale d'environ 2.5 mm et temporelle de 300 à 400 Hz. Quatre types de retours tactiles sont utilisés : le retour tactile à matrice d'aiguilles, le retour vibro-tactile, le retour tactile pneumatique et le retour tactile thermique. D'autres technique telles que les retours électro-tactile, qui consistent à envoyer des impulsions électriques sur la peau ou la stimulation neuromusculaire qui envoie un signal directement au cortex primaire de l'utilisateur sont jugées trop dangereuses par leurs auteurs [6, p82–90]. Toutefois, la société Praxim a développé un prototype constitué d'une matrice de 144 électrodes placées en contact avec la surface de la langue et qui permettrait de se substituer à la vision pour les chirurgiens.

**Retour vibro-tactile :** Les contacts avec la peau se font à une fréquence d'environ 200Hz grâce des barrettes de microtiges mises en mouvement par des bobinages audio. Un système de contrôle fait varier l'amplitude du signal électrique traversant chaque bobinage. Le dispositif traduit ainsi l'amplitude du signal entrant en force de pression sur la peau de l'individu.

**Retour tactile pneumatique :** On utilise un gant muni de micro-ballons à l'endroit où doit être appliquée une pression. Une unité de contrôle active des vannes électropneumatiques qui gonflent ou dégonflent les poches d'air. On améliore la sensation de toucher en augmentant la quantité de ballons utilisés. De

plus, certains types de sensations telles que le glissement nécessitent plusieurs ballonnets en contact avec une portion de surface de peau donné.

### 1.4.2 Retour d'effort

La téléopération à retour d'effort, consiste à faire ressentir à l'opérateur distant les forces qui s'appliquent sur l'outil de travail situé à l'autre bout de la chaîne. Cela permet de se sentir directement en contact avec l'environnement du site opératoire et de gérer aussi les forces que l'on applique sur celui-ci, la chaîne de transmission devenant ainsi quasiment transparente. Pour cela, des capteurs de force sont fixés sur le robot en bout de chaîne et envoient au site maître les valeurs des forces que subit le manipulateur esclave. Le dispositif de commande reproduit ces forces à un facteur d'échelle près, sur le joystik de l'opérateur.

## 1.5 Conclusion

Une des caractéristiques de la plupart des dispositifs de téléopération est l'emploi dans leurs site esclave de robots manipulateur à cause de leur morphologie proche de celle de l'homme. En ce qui concerne le site de capture de mouvements, il reste assez basique—un joystik avec retour d'effort dans le meilleur des cas—en dépit de la disponibilité sur le marché de sytèmes de capture fiables et performants.

On va s'intérresser dans notre projet à l'apport de ces nouveaux systèmes de capture (magnétique et optique) à la téléopération.

# 2

# SYSTÈMES EXPÉRIMENTAUX DE L'INRIA RHÔNE-ALPES

L'INRIA Rhône-Alpes, dispose d'un robot manipulateur industriel 6 axes de la société Staubli, (voir figure 2.1) ainsi que de deux dispositifs de capture de mouvements, l'un optique, l'"Optotrak" de Northern Digital, l'autre magnétique, le "Flock of Birds" de Ascension Technology Corp. et d'une souris à six degrés de liberté. La liaison entre un site de capture de mouvement et le robot Rx90 se fait via le réseau informatique de l'INRIA. Le but de l'opération est de faire reproduire au robot le mouvement de la main d'un opérateur situé dans une salle d'acquisition. Comme le montre le schéma figure 2.1 suivant, on pourra choisir d'utiliser l'une des trois méthodes de capture pour commander le robot.

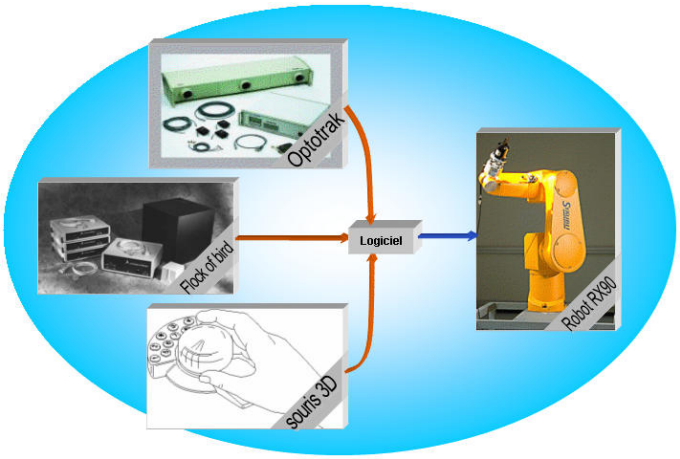



FIG. 2.1 – Schéma de la téléopération

## 2.1 Robot Manipulateur Rx90

### 2.1.1 Caractéristiques

Le manipulateur Rx90 est un robot industriel à 6 degrés de liberté, 6 articulations rotoïdes. Il pèse 108 Kg, le centre de son poignet, c'est-à-dire la cinquième articulation peut se mouvoir dans une zone de travail de rayon compris entre 290 et 900 mm et prenant son origine au centre de la deuxième articulation comme le montre la figure 2.2. Le robot peut atteindre des vitesses de déplacement de 1,5 m/s et à vitesse nominal, il a une répétabilité de +/- 0.02 mm, à température constante, il peut transporter une charge d'environ 6 Kg.

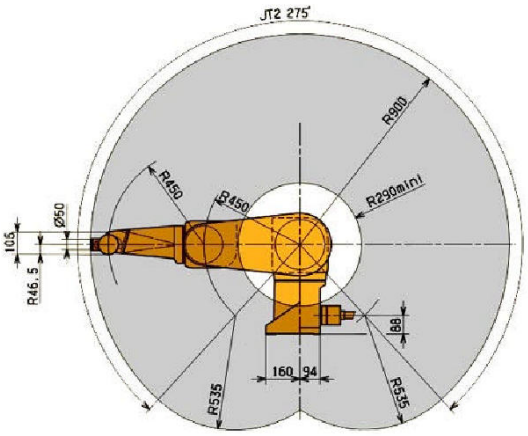



FIG. 2.2 – Zone de travail du robot Rx90

### 2.1.2 Système de commande

La baie de commande se base sur une architecture utilisant comme système d'exploitation VxWorks et qui est relié au réseau de l'INRIA. La commande et l'asservissement du robot Rx90 ont été réalisés à l'INRIA par Roger Pissard-Gibollet, puis une surcouche logicielle appelé RobControl implémentée par Hervé Mathieu, constituée d'un ensemble fonctions en langage C, va nous permettre de disposer de commandes évoluées pour se connecter au robot et le contrôler à distance.

- **int** robctlConnect(RX90) *établit la connexion avec la baie de commande du robot.*
- **int** robctlInit(RX90, 0) *initialise le contrôleur et met sous tension le robot.*
- **int** robctlClose(RX90) *met hors tension le robot.*
- **int** robctlDisconnect(RX90) *termine la connexion avec la baie de commande du robot.*

- **int** robctl(RX90, PUT_POSITION, **(void*)** joint[6]) *envoie au robot la nouvelle consigne en positions articulaires* joint.
- **int** robctl(RX90, WAIT_END_MOVE, **(void*)** NULL) *attend que le robot termine son mouvement pour rendre la main.*
- **int** robctl(RX90, READ_POSITION, **(void*)** joint[6]) *lit les valeurs articulaires du robot et les stocke dans* joint.
- **int** rx90MGD(**double** snaP[3][4], **double** joint[6]) *renvoie dans* snaP *la configuration de l'outil terminal en fonction de la configuration articulaire* joint.
- **int** rx90MGI(**double** snaP[3][4], **double** joint[6], **double** old_joint[6]) *renvoie dans* joint *les valeurs que doivent prendre les articulations pour que l'outil du robot atteigne la configuration donnée par la matrice homogène* snaP, *en tenant compte de la configuration précédente du robot* old_joint *(dans le cas de solutions multiples).*
- **void** rx90JAC(**double** JAC[6][6], **double** joint[6]) *calcule la jacobienne* JAC *en fonction de la configuration articulaire* joint.

## 2.2 Capture de mouvement

Les dispositifs de capture de mouvement doivent envoyer en temps réel les commandes de l'opérateur au dispositif robotique. En ce qui concerne l'"Optotrak" et le "Flock of Birds", il s'agit de reproduire sur le robot Rx90 le mouvement de la main de l'opérateur. La souris 3D, sert à l'utilisateur pour déplacer et orienter l'outil du robot.

### 2.2.1 Optotrak

L'"Optotrack" est constitué de trois caméras et leur objectifs montés sur un bâti linéaire rigide long de 1m10 et plusieurs leds connectées à un strober (collecteur de led). Les leds placées sur le corps de l'individu en mouvement sont allumées successivement. Chaque caméra fournit au calculateur les coordonnées x, y de chaque led lui permettant ainsi de retrouver la position spatiale par stéréoscopie. Ces positions sont données soit par rapport à un repère lié au bâti des cameras, soit par rapport à un "rigid body" constitué de trois leds montés sur un bâti triangulaire. Les positions sont données à une fréquence de 600 Hz.

### 2.2.2 Flock of Birds

Le "Flock of Bird" est un système de capture de mouvement magnétique. Il fournit à une fréquence d'environ 144 Hz la position cartésienne et l'orientation d'un capteur. La zone d'acquisition d'un diamètre d'environ 3 m doit être dégagée de tout objet métallique.

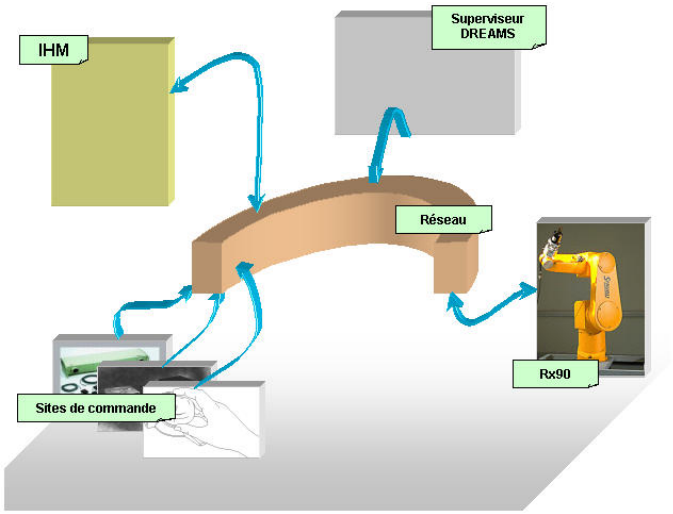
### 2.2.3 Souris 3D

La souris 3D à 6 degrés de liberté. Sa partie supérieure peut se déplacer selon les trois axes x, y et z et tourner autour de chacuns d'eux. Elle fournit ensuite l'amplitude de ces mouvements. Un pilote de périphérique permet de recupérer ses données sur le port série de l'ordinateur auquel la souris est connecté.

# 3

# SPÉCIFICATIONS

## 3.1 Architecture générale




On considére IHM—Interface Homme Machine—qui permet de contrôler le fonctionnement de la téléopération. Par exemple, l'opérateur pourra y choisir quel organe de capture de mouvement il souhaite utiliser ainsi que définir certaines proprietés de la téléopération. Elle devra en outre être accessible depuis n'importe quel poste connecté au réseau.

Le superviseur robotique DREAMS va servir dans un premier temps pour commander la mise en marche et l'arret de la téléopération.

Les dispositifs de capture de mouvements sont connectés au réseau et emmettent leurs données selon les requêtes faites à travers l'IHM.

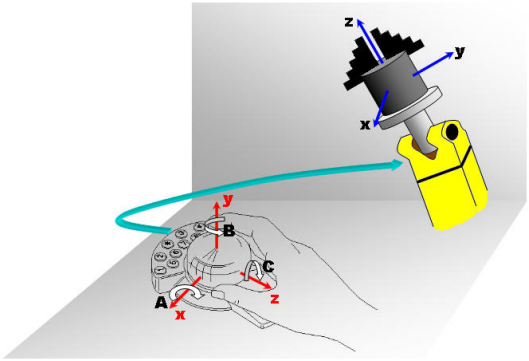
L'armoire de commande du robot Rx90 est connectée au réseau. Une API (Application Protocol Interface) de commande (voir 2.1.2), permet de lui envoyer les ordres de positions de configuration.

## 3.2 Description fonctionnelle

Il serait interessant de doter le bras de l'opérateur de plusieurs marqueurs et de s'interesser à son mouvement. Pour des raisons de temps, l'étude se limitera au mouvement de la main.

Le fonctionnement de la téléopération, c'est-à-dire la manière dont le robot doit rejouer le mouvement de l'opérateur est propre et varie selon le dispositif de capture utilisé.

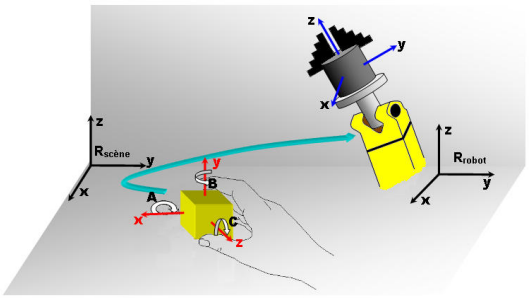
### 3.2.1 Commande par la souris 3D




La souris 3D se branche sur le port série d'une station linux. Un driver capture et met à disposition les paramètres de déplacement de son organe mobile.

Le mouvement du poignet du Rx90 doit suivre le mouvement appliqué à la souris 3D. Si l'organe mobile de cette dernière tourne ou se déplace par rapport à un axe, par exemple z, le poignet du robot doit en faire autant.

### 3.2.2 Commande par l'Optotrak ou le Flock of Birds



L'opérateur tient le marqueur du dispositif de capture de mouvement dans sa main. La position et l'orientation du repère lié au marqueur, sont données par rapport à un repère fixe par rapport à la scène d'acquisition.

## 3.3 Performances à atteindre

### 3.3.1 Types de reproduction de mouvements

Le logiciel de téléopération développé doit offrir la possibilité à l'opérateur de suspendre la téléopération en cours via l'IHM. Lorsque l'opérateur remettra en marche la téléopération, si le marqueur s'est déplacé durant la suspension, ce déplacement n'est pas reporté sur le robot. Par contre si l'opérateur déplace le marqueur aprés la remise en marche, seul ce déplacement est reporté, et ceci sans tenir compte du fait que la position de départ à changé. On peut illustrer ce type de fonctionnement par une analogie avec une souris d'ordinateur et son pointeur sur l'écran.

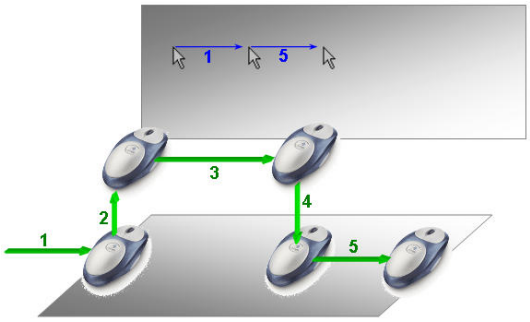



FIG. 3.1 – Seul le mouvement est reporté lorsque la souris est posée, et non la position

En effet lorsque l'on déplace la souris, seul le mouvement est reporté et non la position de la souris par rapport au tapis. Si on la décolle de ce dernier, le pointeur ne se déplace plus sur l'écran jusqu'à ce qu'elle soit reposée. Si par la suite on la déplace, le pointeur part de sa position courante sans tenir compte du fait que la souris n'est pas à la même place.

### 3.3.2 Démultiplication de mouvements

L'environnement de travail du robot et celui de l'opérateur n'ont pas forcément la même dimension. L'opérateur peut travailler à distance sur des environnements beaucoup plus grands que son espace d'acquisition. Pour faciliter, voire permettre de telles opérations, on a recours à l'utilisation d'un coefficient d'échelle entre les mesures prises auprès de l'opérateur et celles reportées sur le robot. Dans notre cas, l'IHM permet de spécifier deux paramètres, l'un est le

coefficient de démultiplication des distances entre la zone d'acquisition et la zone de travail du robot et l'autre, un coefficient de démultiplication de l'amplitude de la réorientation.

La partie suivante présente la conception et la réalisation de ces spécifications

# 4

# CONCEPTION ET IMPLÉMENTATION D'UNE MÉTHODE D'INTERFAÇAGE

## 4.1 Environnement logiciel

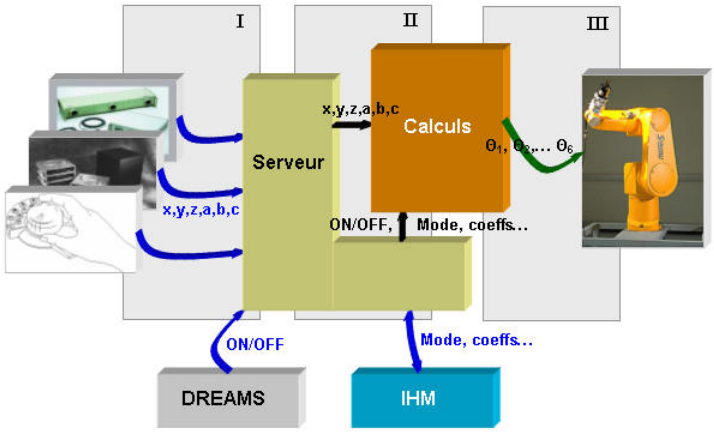



FIG. 4.1 – Logiciel

Trois grandes parties se distinguent dans la conception du logiciel :
- **Partie I** Protocoles de communication et traitement des informations avant émission.
- **Partie II** Transformation des données capturées en commandes pour le robot.
- **Partie III** Commande du robot.

## 4.2 Liaison avec les dispositifs de capture (Partie I)

On peut diviser le traitement des données capturées en quatre blocs indépendants (figure 4.2). Le premier bloc—Reconstruction—est propre à chaque dispositif de capture et les trois autres sont identiques pour tous.

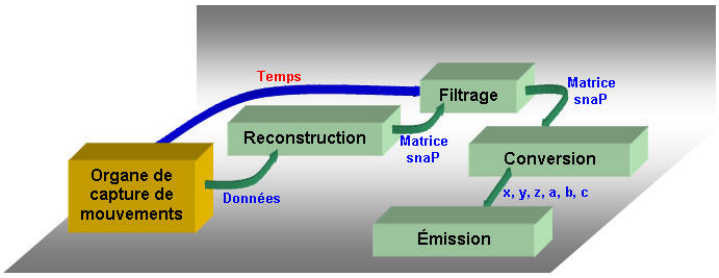



FIG. 4.2 – Chaîne de traitement des données

### 4.2.1 Construction du repère lié à l'individu

#### 4.2.1.1 Optotrak

Le module attaché à l'opérateur est une plaque rigide contenant six marqueurs (diodes infrarouges) auquel on associe un repère $R_{capt}$ (voir figure 4.3).

L'Optotrak fournit à une cadence de 100 Hz la position de chacun des capteurs par rapport au repère $R_{opto}$ lié à la scène (figure 4.3).

Pour déduire des positions des capteurs, la configuration du repère mobile ainsi que l'orientation de ses axes, on utilise une identification au sens des moindres carrés.

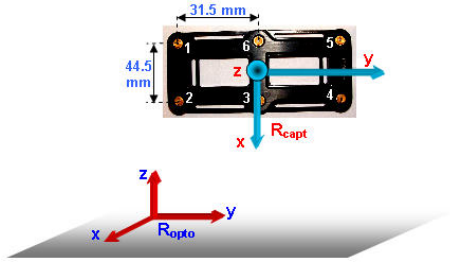



FIG. 4.3 – Plaque de maqueurs

On dispose alors des coordonnées cartésiennes $P_{capt_i} = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}_{/R_{capt}}$ de chaque capteur $i$ par rapport au repère $R_{capt}$ (voir mesures de la figure 4.3).

Par ailleur, l'Optotrak nous fournit la position $P'_{capt_i} = \begin{pmatrix} P'_x \\ P'_y \\ P'_z \end{pmatrix}_{/R_{opto}}$ par rapport au repère lié à la scène d'acquisition.

Il reste alors à déterminer la position du repère $R_{capt}$ par rapport au repère $R_{opto}$, ce qui revient à exprimer la matrice de passage ${}^{R_{opto}}T_{R_{capt}}$. On a la relation suivante :

$$ {}^{R_{opto}}T_{R_{capt}}(P_{capt_i})_{/R_{capt}} = (P'_{capt_i})_{/R_{opto}} $$

soit :

$$ \begin{bmatrix} sx & nx & ax & P''_x \\ sy & ny & ay & P''_y \\ sz & nz & az & P''_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{x_i} \\ P_{y_i} \\ P_{z_i} \\ 1 \end{bmatrix} = \begin{bmatrix} P'_{x_i} \\ P'_{y_i} \\ P'_{z_i} \\ 1 \end{bmatrix} $$

On obtient trois systèmes de six equations qui une fois résolus permettent de calculer une approximation de la valeur des éléments de la matrice homogéne.

#### 4.2.1.2 Souris 3D

Lorsque l'opérateur utilise la souris 3D, cette dernière lui fournit l'amplitude de son mouvement sous la forme de trois valeurs notant le déplacement selon les axes x, y et z et trois angles nautiques $\Phi$, $\theta$ et $\Psi$. Au démarrage, une matrice homogène traduisant la position d'un repère dans l'espace est initialisée à l'identité. Lorsque le joystick de la souris est déplacé, cette matrice est remise à jour de telle façon que le repère correspondant pivote ou se déplace de la même manière. Ainsi, ce qui est envoyé ce sont les positions dans le temps d'un repère virtuel.

#### 4.2.1.3 Flock of Birds

Le "Flock of Birds" donne directement la position et l'orientation dans l'espace du repère lié au capteur porté par l'opérateur. L'opération de reconstruction se limite à transformer ces coordonnées de la forme position-angles nautiques en une matrice de cosinus directeur équivalente.

### 4.2.2 Traitement et émission des données

#### 4.2.2.1 Filtrage spatial et temporel

L'émission des données se fait via le réseau informatique de l'INRIA Rhônes-Alpes. Il est nécessaire d'optimiser sur ce genre de médium la quantité d'informations émises. Pour cela toutes les données de position issues des dispositifs de capture sont d'abord filtrées.

Le filtrage considère trois paramètres : le temps écoulé, la distance spatiale et la réorientation entre deux points successifs émis. Ainsi, un point capturé ne sera émis que s'il s'est écoulé, depuis la dernière émission, un temps supérieur à une valeur fixée ou si la distance ou la réorientation par rapport au dernier point émis est supérieure à la résolution demandée par l'utilisateur.

#### 4.2.2.2 Protocole de communication

Les opérations décrites précédemment effectuent des calculs avec des points exprimés en matrice homogène. Ces matrices à 12 éléments (si on néglige le dernière ligne qui est toujours [0 0 0 1]) sont redondantes. En effet, on peut relever certaines relations mathématique liant les éléments entre eux, par exemple : $\|\vec{s}\| = 1$, $\|\vec{n}\| = 1$, $\|\vec{a}\| = 1$ et $\vec{s} \wedge \vec{n} = \vec{a}$.

Ainsi pour éviter de transmettre des information redondantes, les matrices homogène sont transformées en positions et angles nautiques.

## 4.3 Gestion de la téléopération (Partie II)

La partie II (voir section 4.1 page 23), s'occupe d'interpréter les données de positions venant des dispositifs de capture de mouvement pour générer les ordres de trajectoires du robot selon les paramètres spécifiés dans l'IHM. On trouvera en annexe le schéma de l'implémentation de cette partie ainsi que le fichier de programmation en langage C.

### 4.3.1 Modes de reproduction de mouvements

La principale spécification pour effectuer une téléopération est la façon dont le logiciel doit reporter les positions du repère maître de la scène d'acquisition dans l'espace de travail du robot. On distingue ici trois types de report.

#### 4.3.1.1 Report des positions et des orientations par rapport aux repères fixes

La position du poignet de l'opérateur en salle d'acquisition, données par rapport à un repère fixé à la scène (soit un corps rigide [1]) ou le repère propre du système de capture) sera reporté sur le repére de base du robot.

Ce mode a très peu d'intérêt car les deux environnements ne sont en général pas disposés de la même manière par rapport à leur repère respectif. Si l'utilisateur veut atteindre un point situé à 3cm du repère du robot, il devra amener son capteur de position à 3 cm du repère lié à la scène d'acquisition. Or bien souvent, du fait de la disposition de cette scène et de la position de ce repère, cette position est inatteignable.

#### 4.3.1.2 Report des déplacements et des réorientations par rapport aux repères fixes

Pour supprimer la contrainte posée par la position des repères de référence de chacun des sites, on ne reporte d'un site à l'autre que le déplacement du poignet de l'opérateur. Pour ce faire, la nouvelle position du poignet du robot est déterminée en considérant l'ancienne que l'on translate suivant les axes du

[1] bâti rigide d'au moins 3 marqueurs pouvants servir de repère de référence

repère de base. Les longueurs et les directions de cette translation sont celles constatées entre deux points successifs issus du capteur de mouvement.

Ce mode ne permet de reporter sur le robot que le déplacement de l'opérateur. Si l'opérateur fait une translation selon l'axe x du repère de la salle d'acquisition, le poignet du robot fera aussi une translation selon l'axe x du repère lié au robot. On ne tient pas compte ici des positions exactes mais uniquement des déplacements.

Ainsi en utilisant ce mode, on s'affranchit des contraintes liées à la position des repères de référence. Cependant, on reste tributaire des directions de ces repères. Par exemple, si l'opérateur déplace son outil vers l'avant, et que cette direction correspond à une translation selon l'axe z du repère d'acquisition, l'outil du robot se déplacera lui aussi selon cet axe z de son repère de référence. Or, cet axe z est dirigé dans l'espace de travail du robot vers le haut, l'outil du robot fera un mouvement ascendant contrairement à l'intention de l'utilisateur.

Le raisonnement est analogue pour le report des orientations de l'outil.

#### 4.3.1.3 Report des déplacements et des réorientations par rapport aux repères mobiles

Pour les raisons que l'on a citées ci-dessus, les axes des repères de base ne constituent pas une bonne référence pour les directions de déplacement. Les axes du repère des outils semblent plus appropriés pour repèrer les mouvements. Ainsi, après acquisition d'un nouveau point, on considere les distances de translation par rapport à chaque point précédent puis on reporte ces valeurs selon les mêmes axes du repère outil actuel du robot. On applique la même opération pour reporter la réorientation. (voir le détail des calculs en annexe)

Cette méthode convient mieux pour effectuer des tâches à distance, car elle ne considére que le sens de déplacement de son outil relativement à lui même : par exemple, des déplacements latéraux de celui-ci engendreront des déplacements latéraux de l'outil distant.

### 4.3.2 Opération sur les données

L'espace de travail et celui d'acquisition ne présentent souvent pas les mêmes caractéristiques. Au niveau de l'envergure, il est primordial de pouvoir faire effectuer des mouvements amples au robot même si l'espace d'acquisition ne permet pas de contenir une telle amplitude.

#### 4.3.2.1 Suspension et reprise de la téléopération

La première méthode retenue (voir section 3.3.1 page 24) est basée sur une analogie avec une souris d'ordinateur. En effet, lors d'un mouvement, si la souris arrive au bord du tapis alors que le pointeur sur l'écran n'a pas encore atteint l'endroit désiré, on soulève la souris, on la repositionne au milieu du tapis et on continue notre mouvement.

Le logiciel développé réagit de la même manière : si l'utilisateur arrive au frontières de sa zone d'acquisition, il envoie un ordre de suspension au logiciel qui maintient alors le robot dans la position courante. Apres s'être repositionné, l'utilisateur envoie un ordre de reprise et le mouvement du robot reprend à la position d'arrêt.

Les ordres de suspension et de reprise sont pour l'instant à spécifier à travers l'IHM, mais seront à terme implantés dans un commutateur que l'utilisateur tiendra dans sa main.

#### 4.3.2.2 Démultiplication des mouvements

La démultiplication de mouvement n'est possible que dans les modes où les déplacements et les réorientations sont reportés. Cette opération consiste lors du report d'un déplacement ou d'une réorientation de multiplier celle-ci par un coefficient spécifié à travers l'IHM.

## 4.4 Liaison avec le robot Rx90

### 4.4.1 Passage de l'espace opérationnel à l'espace articulaire

Tous les traitements de données se font dans l'espace opérationnel du robot, c'est-à-dire que l'on considère les positions cartésiennes, l'orientation de son outil terminal dans le repère lié à sa base. Par ailleurs, le déplacement du robot se fait par l'envoi de commande aux moteurs placés sur ses articulations. Ces moteurs agissent sur l'angle et la vitesse angulaire de l'articulation à laquelle ils sont attachés. Il est alors nécessaire pour amener le robot à une position et une vitesse précise de l'espace opérationnel, de déterminer les valeurs angulaires correspondantes qui serviront à la commande des moteurs.

#### 4.4.1.1 Positions

Pour effectuer la correspondance : coordonnées spaciales-coordonnées articulaires, on utilise le modèle géométrique inverse. Cette correspondance n'est pas bijective, c'est-à-dire que pour atteindre une position spatiale le robot peut prendre plusieurs configurations articulaires. Il est alors nécessaire de choisir la plus appropriée au mouvement que l'on veut exécuter. Le critère de choix des positions multiples retenu est la minimisation des distances angulaires des configurations successives. Ainsi, pour passer d'une position à une autre, on choisit la configuration la plus proche de la précédente.

Le modèle géométrique inverse du robot Rx90, développé à l'INRIA Rhône Alpes par Roger Pissard-Gibollet, a été amélioré de façon à optimiser le choix des configurations multiples comme présenté ci-dessus.

#### 4.4.1.2 Vitesses

Nous connaissons le temps que doit mettre le robot pour passer d'une position à une autre, donc d'une configuration articulaire [2] à une autre. La vitesse articulaire se détermine alors aisément en calculant la variation de chaque valeur articulaire et en divisant par le temps.

Toutes ces données—configurations articulaires et vitesses articulaires— servent au contrôleur pour commander le robot.

### 4.4.2 Contrôleur du robot (Partie III)

Une fois les données de positions transformées en une suite de configurations articulaires, il reste alors à les envoyer au contrôleur du robot qui se charge de générer la commande adéquate des moteurs pour obtenir le mouvement désiré. Cette communication est assurée par une API réalisée par Hervé Mathieu (voir section 2.1.2 page 19) puis modifiée pour tenir compte des nouvelles contraintes developpées ci-après.

#### 4.4.2.1 Elimination des points d'arrêt

Une trajectoire est constituée d'une suite de points. Grâce au modèle géométrique inverse du robot, on détermine les configurations articulaires nécessaires pour atteindre chacun de ces points. Ces configurations sont alors envoyées à la suite au contrôleur du robot. Cependant, la loi de commande originelle suppose que le robot parte d'une position à vitesse nulle et arrive à la position suivante à vitesse nulle. Cela a pour conséquence de marquer sur une trajectoire des temps d'arrêt inutiles qui se traduisent par des mouvements saccadés nuisibles aux articulations.

La première étape consiste alors à éliminer au niveau de la commande ces points d'arrêt en imposant aux articulations du robot, lors du parcours d'une trajectoire, de débuter leur mouvement avec la vitesse articulaire qui a permit d'atteindre le point précèdent et de la terminer à une vitesse spécifiée par l'utilisateur. Au niveau du contrôleur, les polynômes d'interpolation qui fournissent en fonction du temps la valeur des articulations ont été modifiés pour tenir compte de ces contraintes (voir annexe).

[2] ensemble des valeurs prises par les articulations du robot

# 5

# RÉSULTATS EXPÉRIMENTAUX

Beaucoup d'éléments interviennent dans la mise en place de la téléopération. Pour faciliter l'implémentation et le teste, on a divisé le processus comme suis :

- **BLOC I** Conversion des données après acquisition
- **BLOC II** Conversion positions cartésiennes en vecteurs articulaires
- **BLOC III** Commande du robot

Pour tester le bon fonctionnement de toute la chaîne, on a d'abord commencé à tester le bloc III seul puis les blocs II et III assemblés et enfin les 3 blocs ensembles.

## 5.1 Commande du robot

La nouvelle commande du robot RX90 devait supprimer les saccades dans les mouvements du robot (voir chapitre 4). Le tests consiste ici à donner au contrôleur une suite de positions articulaires et d'observer son mouvement.

Les configurations articulaires spécifiées sont atteintes avec un mouvement continu. Les points de la trajectoire ne se distinguent plus du corps de la trajectoire.

## 5.2 Conversion positions cartésiennes en vecteur articulaire

Ce bloc a pour rôle d'extraire d'une suite de points, le déplacement de l'outil d'acquisition et de le transposer en un déplacement dans l'espace de travail du robot. Ensuite il s'agira de traduire ces déplacements en une suite de positions articulaire avant de les envoyer au robot.

Pour tester ce bloc on a généré trois fichiers de points. Le premier contient une suite de points dont l'abscisse à été incrémentée, le second l'ordonnée et le troisième la cote.

Ces trois fichiers sont passés par ce bloc. Il nous fournit pour une suite de configurations articulaires. Ces configurations sont ensuite données comme commande au robot.

On observe le comportement suivant :
- **Fichier I :** Déplacement du poignet selon son axe x
- **Fichier II :** Déplacement du poignet selon son axe y
- **Fichier III :** Déplacement du poignet selon son axe x

Comportements bien conforme aux spécifications. En effet le mouvement se fait bien relativement au repère de l'outil et non par rapport au repère de base.

Pour tester la réorientation, on a généré un fichier de points avec la formule suivante :

$$snaP_i = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & -P_z\sqrt{{P_y}^2 + {P_z}^2} & -P_y\sqrt{{P_y}^2 + {P_z}^2} & P_y \\ 0 & P_y\sqrt{{P_y}^2 + {P_z}^2} & -P_z\sqrt{{P_y}^2 + {P_z}^2} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

avec : $P_x = 0.4$, $P_y = 0.3\cos\frac{i\pi}{18}$ et $P_z = 0.3\cos\frac{i\pi}{18}$

Ces points représentent des repères dont l'origine décrit un cercle dans l'espace et dont l'axe z est toujours orienté vers le centre de celui-ci et l'axe x reste constant.

Durant la manipulation on observe bien que le centre du poignet du robot décrit un cercle et son outil tourne bien autour de l'axe x.

## 5.3 Conversion des données après acquisition

Pour tester la conversion de données, on a effectué une trajectoire rectiligne dams la scène d'acquisition. Les données son récupérées sur fichier après conversion et passées par les deux autres blocs. Le robot décrit bien la trajectoire rectiligne.

## 5.4 conclusion

Les différents blocs de la téléopération fonctionnent ensemble, cependant la liaison entre l'acquisition et les autres blocs n'est pas encore implémentée. Pour finaliser, le projet, il reste à assembler les différentes parties par une communication réseau. Cette dernière étape sera réalisée durant le sixième mois de stage.

Par ailleurs, la téléopération utilisant la souris 3D et ne nécessitant pas une communication réseau est déjà opérationnelle.

# CONCLUSION

Le travail réalisé durant ce stage a pour objectif de concevoir et de mettre en place une solution fiable de téléopération entre plusieurs systèmes de captures de mouvement et un robot manipulateur. La première étape a consisté en une étude des architectures couramment utilisées pour ce genre d'opération. Ensuite le travail s'est focalisé sur la mise en oeuvre et l'implémentation d'une méthode fonctionnel.

Grâce à son aspect innovant, ce projet m'a permis de travailler de manière autonome sans me référer à aucun modéle, c'est-à-dire de déterminer de ma propre initiative en concertation avec mes encadrants les orientations des études à mener et les solutions à utiliser. En dehors des compétences certaines en robotique tant dans l'asservissement que dans la commande haut niveau, ce projet m'a aussi permis d'acquérir une meilleur maîtrise de l'outil informatique ainsi qu'une culture générale sur les principes et techniques liés au domaine de la réalité virtuelle.

# A

# CONTRÔLEUR DU ROBOT : POLYNÔMES D'INTERPOLATION

## A.1 Interpolation par un polynôme de degré 3 :

On cherche à determiner une équation de trajectoire $q(t) = q_i + r(t)D$ avec $D = q_f - q_i$ qui satisfasse les conditions suivantes :

1. Position initiale : $q(t = 0) = q_i$
2. Position finale : $q(t = t_f) = q_f$
3. Vitesse initiale : $\dot{q}(t = 0) = \dot{q}_i$
4. Vitesse finale : $\dot{q}(t = t_f) = \dot{q}_f$

Il faut donc pour satisfaire ces quatres équations, choisir un polynôme degré 3 au minimum. On utilise alors :

$$r(t) = at^3 + bt^2 + ct + d \tag{A.1}$$

les équations précédentes deviennent :

$$d = 0 \tag{A.2}$$
$$at_f^3 D + bt_f^2 D + ct_f D + q_i - q_f = 0 \tag{A.3}$$
$$c - \frac{\dot{q}_i}{D} = 0 \tag{A.4}$$
$$3at_f^2 D + 2bt_f D + cD - \dot{q}_f = 0 \tag{A.5}$$

En résolvant ces équations par rapport aux coefficients du polynôme (A.1), on obtient :

$$a = \frac{t_f\dot{q}_f + 2q_i - 2q_f + t_f\dot{q}_i}{t_f^2 D} \tag{A.6}$$

$$b = -\frac{t_f\dot{q}_f + 3q_i - 3q_f + 2t_f\dot{q}_i}{t_f^2 D} \tag{A.7}$$

$$c = \frac{\dot{q}_i}{D} \tag{A.8}$$

$$d = 0 \tag{A.9}$$

D'où l'expression de $r(t)$ suivante :

$$r(t) = \frac{(t_f\dot{q}_f + 2q_i - 2q_f + t_f\dot{q}_i)t^3}{t_f^3 D} - \frac{(t_f\dot{q}_f + 3q_i - 3q_f + 2t_f\dot{q}_i)t^2}{t_f^2 D} + \frac{\dot{q}_i t}{D} \tag{A.10}$$

En introduisant $\tau = \frac{t}{t_f}$ et $D = q_f - q_i$, il vient :

$$r(\boldsymbol{t}) = 3\boldsymbol{\tau^2} - 2\boldsymbol{\tau^3} + \frac{(\boldsymbol{\tau^3} - \boldsymbol{\tau^2})t_f\dot{q}_f + ((\boldsymbol{\tau^3} - 2\boldsymbol{\tau^2})t_f + \boldsymbol{t})\dot{q}_i}{D} \tag{A.11}$$

ou encore :

$$r(\boldsymbol{\tau}) = 3\boldsymbol{\tau^2} - 2\boldsymbol{\tau^3} + \frac{(\boldsymbol{\tau^3} - \boldsymbol{\tau^2})\dot{q}_f + (\boldsymbol{\tau^3} - 2\boldsymbol{\tau^2} + \boldsymbol{\tau})\dot{q}_i}{D} t_f \tag{A.12}$$

## A.2 Interpolation par un polynôme de degré 5 :

On cherche à determiner une équation de trajectoire $q(t) = q_i + r(t)D$ avec $D = q_f - q_i$ qui satisfasse aux conditions suivantes :

1. Position initiale : $q(t = 0) = q_i$
2. Position finale : $q(t = t_f) = q_f$
3. Vitesse initiale : $\dot{q}(t = 0) = \dot{q}_i$
4. Vitesse finale : $\dot{q}(t = t_f) = \dot{q}_f$
5. Accélération initiale : $\ddot{q}(t = 0) = 0$
6. Accélération finale : $\ddot{q}(t = t_f) = 0$

Il faut donc pour satisfaire ces six equations choisir un polynôme de degré 5 au minimum. On utilise alors :

$$r(t) = at^5 + bt^4 + ct^3 + dt^2 + et + f \tag{A.13}$$

Les équations précédentes deviennent :

$$fd = 0 \tag{A.14}$$
$$q_i - q_f + D(at_f^5 + bt_f^4 + ct_f^3 + dt_f^2 + et_f + f) = 0 \tag{A.15}$$
$$eD - \dot{q}_i = 0 \tag{A.16}$$
$$D(5at_f^4 + 4bt_f^3 + 3ct_f^2 + 2dt_f + e) - \dot{q}_f = 0 \tag{A.17}$$
$$2dD = 0 \tag{A.18}$$
$$D(20at_f^3 + 12bt_f^2 + 6ct_f + 2d) = 0 \tag{A.19}$$

En resolvant ces équations par rapport aux coefficients du polynôme (A.13), on obtient :

$$a = -3\frac{2q_i - 2q_f + t_f\dot{q}_i + t_f\dot{q}_f}{Dt_f^5} \tag{A.20}$$
$$b = \frac{15q_i - 15q_f + 8t_f\dot{q}_i + 7t_f\dot{q}_f}{Dt_f^4} \tag{A.21}$$
$$c = -2\frac{5q_i - 5q_f + 3t_f\dot{q}_i + 2t_f\dot{q}_f}{Dt_f^3} \tag{A.22}$$
$$e = \frac{\dot{q}_i}{D} \tag{A.23}$$
$$f = 0 \tag{A.24}$$
$$d = 0 \tag{A.25}$$

D'où l'expression de $r(t)$ suivante :

$$r(\boldsymbol{t}) = -3\frac{2q_i - 2q_f + t_f\dot{q}_i + t_f\dot{q}_f}{Dt_f^5}\boldsymbol{t^5} + \frac{15q_i - 15q_f + 8t_f\dot{q}_i + 7t_f\dot{q}_f}{Dt_f^4}\boldsymbol{t^4} - 2\frac{5q_i - 5q_f + 3t_f\dot{q}_i + 2t_f\dot{q}_f}{Dt_f^3}\boldsymbol{t^3} + \frac{\dot{q}_i}{D}\boldsymbol{t} \tag{A.26}$$

En introduisant $\tau = \frac{t}{t_f}$ et $D = q_f - q_i$, il vient :

$$r(\boldsymbol{t}) = 6\boldsymbol{\tau^5} + 10\boldsymbol{\tau^3} - 15\boldsymbol{\tau^4} + \frac{(-4\boldsymbol{\tau^3}t_f + 7\boldsymbol{\tau^4}t_f - 3\boldsymbol{\tau^5}t_f)\dot{q}_f + (-6\boldsymbol{\tau^3}t_f + 8\boldsymbol{\tau^4}t_f - 3\boldsymbol{\tau^5}t_f + \boldsymbol{t})\dot{q}_i}{D} \tag{A.27}$$

ou encore :

$$r(\boldsymbol{\tau}) = 6\boldsymbol{\tau^5} + 10\boldsymbol{\tau^3} - 15\boldsymbol{\tau^4} + \frac{(-4\boldsymbol{\tau^3} + 7\boldsymbol{\tau^4} - 3\boldsymbol{\tau^5})\dot{q}_f + (-6\boldsymbol{\tau^3} + 8\boldsymbol{\tau^4} - 3\boldsymbol{\tau^5} + \boldsymbol{\tau})\dot{q}_i}{D}t_f \tag{A.28}$$

# B

# TRANSFORMATION RELATIVE AU REPÈRE MOBILE

On cherche à determiner la transformation $T$ permettant de passer d'une configuration $T_{i_{opto}}$ d'un point de la zone d'acquisition à la configuration $T_{f_{opto}}$ du point suivant. On a alors :

$$T_{i_{opto}} T = T_{f_{opto}}$$

ou encore :

$$\begin{bmatrix} A_{i_{opto}} & P_{i_{opto}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} rot & trans \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A_{f_{opto}} & P_{f_{opto}} \\ 0 & 1 \end{bmatrix}$$

Alors :

$$\begin{bmatrix} A_{i_{opto}} rot & A_{i_{opto}} trans + P_{i_{opto}} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A_{f_{opto}} & P_{f_{opto}} \\ 0 & 1 \end{bmatrix}$$

On obtient :

$$A_{i_{opto}} rot = A_{f_{opto}}$$

et

$$A_{i_{opto}} trans + P_{i_{opto}} = P_{f_{opto}}$$

d'où :

$$rot = {A_{i_{opto}}}^{-1} A_{f_{opto}}$$

et

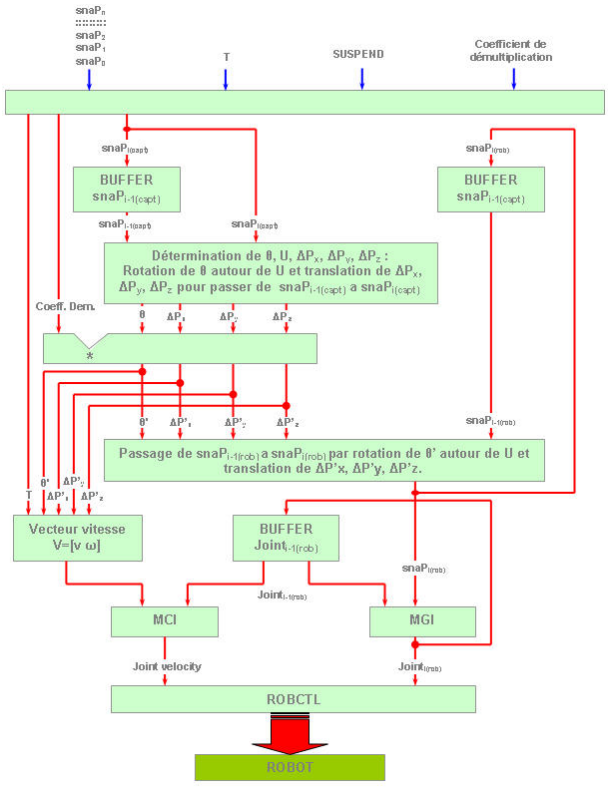
$$trans = {A_{i_{opto}}}^{-1} (P_{f_{opto}} - P_{i_{opto}})$$

Cette matrice de passage $T$ sert pour passer d'un point à l'autre dans le repere du robot.

# C

# GESTION DE LA TÉLÉOPÉRATION (PARTIE II)

## C.1 Schèma



## C.2 Programme source en langage C

```
#include "Rx90Teleop.h"

#define __MCI_NO
#define __NDEBUG_NO

void rx90Commande(double snaP[3][4], double T, int SUSPEND, double
C_Vel, double C_Pos, int BEGIN , int ABS_POS, int ABS_ORI, double
joint_rob_vel[6], double joint_rob_f[6] )
{
  /* — Variables — */
  static double snaP_rob_i[3][4], snaP_opto_i[3][4], joint_rob_i[6];
  double snaP_rob_f[3][4], U[3], rot[3][4], theta, joint[6];
  int i,j,k;

  if( BEGIN == 1 )
    {
#ifdef __NDEBUG
      /* — On met la position 90 20 110 10 10 10 — */
      joint_rob_i[0]= 90.0;
      joint_rob_i[1]= 20.0;
      joint_rob_i[2]= 110.0;
      joint_rob_i[3]= 10.0;
      joint_rob_i[4]= 10.0;
      joint_rob_i[5]= 10.0;
      printf("Eh oui");
#else
      /* — On recupere les joints du robot — */
      robctl(RX90, READ_POSITION, (void *)joint_rob_i);
#endif
      /* — On les transforme en radiants — */
      for(i=0;i<6;i++)
        joint_rob_i[i] = DEGTORAD( joint_rob_i[i] );

      /* — On en deduit sa configuration spaciale snaP — */
      rx90MGD(snaP_rob_i, joint_rob_i);

      /* — Sauvegarde de la position initiale de l'opto — */
      for(i=0;i<3;i++)
        for(j=0;j<4;j++)
          snaP_opto_i[i][j] = snaP[i][j];

      /* — On renvoie la position initiale en radian — */
      for(i=0;i<6;i++)
```

```
        joint_rob_f[i] = joint_rob_i[i];

    }
  else
    {
      switch( ABS_POS + 3*ABS_ORI )
        {
        case 0 :
          /* — ABS_POS = 0 & ABS_ORI = 0 — */
          /* — 1. Calcul de theta et de U (passage de sna_opto_ini a
sna_opto_final)— */
          theta = rx90snaP_to_theta(snaP_opto_i, snaP, U);
          /* — 2. Deduction du passage de sna_rob_ini a sna_rob_final —
*/
          rx90theta_to_snaP( U, theta*C_Pos, rot);
          /* — 3. Calcul de la position rob finale en orientations — */
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)
              {
                snaP_rob_f[i][j] = 0.0;
                for(k=0;k<3;k++)
                  snaP_rob_f[i][j] += snaP_rob_i[i][k] * rot[k][j];
              }

          /* — 4. Puis en position — */
          snaP_rob_f[0][3] = snaP_rob_i[0][3] + (snaP[0][3] -
snaP_opto_i[0][3])*C_Pos;
          snaP_rob_f[1][3] = snaP_rob_i[1][3] + (snaP[1][3] -
snaP_opto_i[1][3])*C_Pos;
          snaP_rob_f[2][3] = snaP_rob_i[2][3] + (snaP[2][3] -
snaP_opto_i[2][3])*C_Pos;
          break;

        case 1 :
          /* — ABS_POS = 1 & ABS_ORI = 0 — */
          /* — 1. Calcul de theta et de U (passage de sna_opto_ini a
sna_opto_final)— */
          theta = rx90snaP_to_theta(snaP_opto_i, snaP, U);
          /* — 2. Deduction du passage de sna_rob_ini a sna_rob_final —
*/
          rx90theta_to_snaP( U, theta*C_Pos, rot);
          /* — 3. Calcul de la position rob finale en orientations — */
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)
              {
                snaP_rob_f[i][j] = 0.0;
```

```
                for(k=0;k<3;k++)
                  snaP_rob_f[i][j] += snaP_rob_i[i][k] * rot[k][j];
              }
          /* — 4. Puis position en absolue — */
          snaP_rob_f[0][3] = snaP[0][3];
          snaP_rob_f[1][3] = snaP[1][3];
          snaP_rob_f[2][3] = snaP[2][3];
          break;

        case 3 :
          /* — ABS_POS = 0 & ABS_ORI = 1 — */
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)
              snaP_rob_f[i][j] = snaP[i][j];
          snaP_rob_f[0][3] = snaP_rob_i[0][3] + (snaP[0][3] -
snaP_opto_i[0][3])*C_Pos;
          snaP_rob_f[1][3] = snaP_rob_i[1][3] + (snaP[1][3] -
snaP_opto_i[1][3])*C_Pos;
          snaP_rob_f[2][3] = snaP_rob_i[2][3] + (snaP[2][3] -
snaP_opto_i[2][3])*C_Pos;
          break;

        case 4 :
          /* — ABS_POS = 1 & ABS_ORI = 1 — */
          for(i=0;i<3;i++)
            for(j=0;j<4;j++)
              snaP_rob_f[i][j] = snaP[i][j];
          break;

        case -1 :
          /* — ABS_POS = -1 & ABS_ORI = 0 — */
          /* — 1. Calcul de theta et de U (passage de sna_opto_ini a
sna_opto_final)— */
          theta = rx90snaP_to_theta(snaP_opto_i, snaP, U);

          /* — 2. Deduction du passage de sna_rob_ini a sna_rob_final —
*/
          rx90theta_to_snaP( U, theta*C_Pos, rot);

          /* — 4. Partie translation — */
          for(j=0;j<3;j++)
            {
              rot[j][3]=0;
              for(i=0;i<3;i++)
                rot[j][3] += snaP[i][j]*(snaP[i][3] -
snaP_opto_i[i][3])*C_Pos;
```

```
            }

          /* — 3. Calcul de la position rob finale en orientations et position
— */
          for(i=0;i<3;i++)
            for(j=0;j<4;j++)
              {
                snaP_rob_f[i][j] = 0.0;
                for(k=0;k<3;k++)
                  snaP_rob_f[i][j] += snaP_rob_i[i][k] * rot[k][j];
              }
          snaP_rob_f[0][3] += snaP_rob_i[0][3];
          snaP_rob_f[1][3] += snaP_rob_i[1][3];
          snaP_rob_f[2][3] += snaP_rob_i[2][3];
          break;

        case 2 :
          /* — ABS_POS = -1 & ABS_ORI = 1 — */
          /* — On recopie l'orientation — */
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)
              snaP_rob_f[i][j]=snaP[i][j];

          /* — 4. On calcul la translation — */
          for(j=0;j<3;j++)
            {
              rot[j][3] = 0.0;
              for(i=0;i<3;i++)
                rot[j][3] += snaP[i][j]*(snaP[i][3] -
snaP_opto_i[i][3])*C_Pos;
            }

          /* — 3. Calcul de la position rob finale en position — */
          for(i=0;i<3;i++)
            {
              snaP_rob_f[i][3] = 0.0;
              for(j=0;j<3;j++)
                snaP_rob_f[i][3] += snaP_rob_i[i][j] * rot[j][3];
            }
          snaP_rob_f[0][3] += snaP_rob_i[0][3];
          snaP_rob_f[1][3] += snaP_rob_i[1][3];
          snaP_rob_f[2][3] += snaP_rob_i[2][3];
          break;
        }

      /* **** MOUVEMENT **** */
```

```
        /* — 1. Calcul du MGI — */
        for(i=0;i<6;i++)
          joint[i]=joint_rob_i[i];
        if( rx90MGI( snaP_rob_f, joint_rob_f, joint) == FALSE )
        {
          printf("\nLimite zone de travail Robot");
          for(i=0;i<6;i++)
            joint_rob_f[i]=joint_rob_i[i];
        }
        else
        {
          rx90BestJoint( joint_rob_f, joint_rob_i );
        }
#ifdef __MCI
        /* — 2. Calcul du vecteur vitesse "v" de V_rob = [v w] — */
        V_rob[0] = (snaP_rob_f[0][3] - snaP_rob_i[0][3]) /T*C_Vel;
        V_rob[1] = (snaP_rob_f[1][3] - snaP_rob_i[1][3]) /T*C_Vel;
        V_rob[2] = (snaP_rob_f[2][3] - snaP_rob_i[2][3]) /T*C_Vel;
        /* — 3. Calcul du vecteur vitesse "w" de V_rob = [v w] — */
        theta = rx90snaP_to_theta(snaP_rob_i, snaP_rob_f, U);
        V_rob[3] = (theta/T)*C_Vel*U[0];
        V_rob[4] = (theta/T)*C_Vel*U[1];
        V_rob[5] = (theta/T)*C_Vel*U[2];
        /* — 4. Calcul du MCI — */
        rx90MCI(joint_rob_i, V_rob, joint_rob_vel);
#else
        /* — Calcul du MCI par une methode plus simple — */
        for(i=0;i<6;i++)
          joint_rob_vel[i] = fabs(joint_rob_f[i] - joint_rob_i[i]) /T*C_Vel;
#endif

        /* **** REINITIALISATION pour le prochain point **** */
        for(i=0;i<6;i++)
          joint_rob_i[i] = joint_rob_f[i];

        for(i=0;i<3;i++)
          for(j=0;j<4;j++)
            {
              snaP_rob_i[i][j] = snaP_rob_f[i][j];
              snaP_opto_i[i][j] = snaP[i][j];
            }
        /* ********************************************** */
    }
}

void rx90MCI(double joint_i[6], double Vel[6], double joint_Vel[6])
```

```
{
  /* — Variables — */
  double JAC[6][6], jac[36], jac_inv[36];
  int i,j,ok;

  /* — 1. Calcul de la jacobienne — */
  rx90JAC(JAC, joint_i);
  for(i=0;i<6;i++)
    for(j=0;j<6;j++)
      jac[i*6+j]=JAC[i][j];

  /* — 2. Inverse de la jacobienne — */
  ok=invert(jac,6,6,jac_inv);

  /* — 3. Calcul des vitesses articulaires — */
  for(i=0;i<6;i++)
    {
      joint_Vel[i] = 0.0;
      for(j=0;j<6;j++)
        joint_Vel[i] += jac_inv[i*6 + j]*Vel[j];
    }
}

double rx90snaP_to_theta(double snaPi[3][4], double snaPf[3][4], double
U[3])
{
  /* — Variables — */
  double s_theta, c_theta, theta, rot[3][3];
  int i,j,k;

  /* — 1. Calcul de la matrice de rotation rot = transpose(snai) * snaf — */
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      {
        rot[i][j] = 0.0;
        for(k=0;k<3;k++)
          rot[i][j] += snaPi[k][i] * snaPf[k][j];
      }

  /* — 2. Calcul de l'angle de rotation autour d'un axe fixe — */
  s_theta = (sqrt( (rot[2][1]-rot[1][2])*(rot[2][1]-rot[1][2]) +
(rot[0][2]-rot[2][0])*(rot[0][2]-rot[2][0]) + (rot[1][0]-rot[0][1])*(rot[1][0]-rot[0][1])
))/2;
  c_theta = ( rot[0][0] + rot[1][1] + rot[2][2] -1 ) /2;
  theta = atan2( s_theta , c_theta );
```

```
  /* — 3. Calcul des axes du vecteur de rotation — */
  if(theta == 0.0)
    {
      U[0]=1.0;
      U[1]=0.0;
      U[2]=0.0;
    }
  else
    {
      U[0] = ( rot[2][1] - rot[1][2] ) < 0.0? -1.0 : 1.0;
      U[0] = U[0] * sqrt( fabs(( rot[0][0] - c_theta )/( 1 - c_theta )) );

      U[1] = ( rot[0][2] - rot[2][0] ) < 0.0? -1.0 : 1.0;
      U[1] = U[1] * sqrt( fabs(( rot[1][1] - c_theta )/( 1 - c_theta )) );

      U[2] = ( rot[1][0] - rot[0][1] ) < 0.0? -1.0 : 1.0;
      U[2] = U[2] * sqrt( fabs(( rot[2][2] - c_theta )/( 1 - c_theta )) );
    }
  return(theta);

}

void rx90theta_to_snaP(double U[3], double theta, double rot[3][4])
{
  rot[0][0] = U[0]*U[0]*(1-cos(theta)) + cos(theta);
  rot[1][0] = U[0]*U[1]*(1-cos(theta)) + U[2]*sin(theta);
  rot[2][0] = U[0]*U[2]*(1-cos(theta)) - U[1]*sin(theta);

  rot[0][1] = U[1]*U[0]*(1-cos(theta)) - U[2]*sin(theta);
  rot[1][1] = U[1]*U[1]*(1-cos(theta)) + cos(theta);
  rot[2][1] = U[1]*U[2]*(1-cos(theta)) + U[0]*sin(theta);

  rot[0][2] = U[2]*U[0]*(1-cos(theta)) + U[1]*sin(theta);
  rot[1][2] = U[2]*U[1]*(1-cos(theta)) - U[0]*sin(theta);
  rot[2][2] = U[2]*U[2]*(1-cos(theta)) + cos(theta);

}

void rx90RTL_to_sna(double phi, double theta, double psi, double
rot[3][3])
{
  rot[0][0] = cos(phi)*cos(theta);
  rot[1][0] = sin(phi)*cos(theta);
  rot[2][0] =-sin(theta);
  rot[0][1] = cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi);
  rot[1][1] = sin(phi)*sin(theta)*sin(psi) + cos(phi)*cos(psi);
```

```
  rot[2][1] = cos(theta)*sin(psi);
  rot[0][2] = cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi);
  rot[1][2] = sin(phi)*sin(theta)*cos(psi) - cos(phi)*sin(psi);
  rot[2][2] = cos(theta)*cos(psi);

}

int rx90Close()
{
  int ret;
  if (robctlClose(RX90)==ERROR)
    return(ERROR);

  ret = robctlDisconnect(RX90);
  return(ret);
}

int rx90Boot()
{
  int ret;
  /*Boot the Rx90 Controller*/
  system("xterm -T test -n test -bg Orange -e rsh affligem
/local/projets/robotique/wind/host/sun4-solaris2/bin/windsh -s
/local/projets/robotique/src/Robctl/Rrobctl-rx.sh lambic &" );
  /*VERIFIER LE TEMPS D'ATTENTE*/
  sleep(5);

  /* Connection to the Rx90 Controller */
  ret = robctlConnect( RX90 );
  /* Put the power on the Rx90 Controller */
  if (ret==OK)
    ret = robctlInit(RX90, 0);

  return(ret);
}

void rx90BestJoint( double joint[6], double oldjoint[6] )
{
  double theta;

  if( fabs(joint[3]) >=0 )
    {
      theta = joint[3] < 0 ? joint[3]+2*Pi : joint[3]-2*Pi;
      if( fabs(joint[3] - oldjoint[3]) >= fabs(theta - oldjoint[3]) )
        joint[3] = theta;
    }
```

```
  if( fabs(joint[5]) >=0 )
    {
      theta = joint[5] < 0 ? joint[5]+2*Pi : joint[5]-2*Pi;
      if( fabs(joint[5] - oldjoint[5]) >= fabs(theta - oldjoint[5]) )
        joint[5] = theta;
    }
}

void rx90Move( double joint[6], double joint_vel[6], double T)
{
  int i,ret,bk,tmp;
  double q[13];

  for(i=0;i<6;i++)
    {
      q[i] = RADTODEG(joint[i]); /*— Position —*/
      q[i+6] = RADTODEG(joint_vel[i]); /*— Velocity —*/
    }
  q[12]= T; /*— Time —*/

  /*— On affiche positions et vitesses —*/
  printf("\n position -->");
  for(i=0;i<6;i++)
    printf("%lf ",q[i]);
  printf("\n vitesse -->");
  for(i=6;i<12;i++)
    printf("%lf ",q[i]);

  /*— Saturation des vitesses a 28 degres par seconde —*/
  for(i=6;i<12;i++)
    if(q[i]>28)
      {
        q[i]=28.0;
        q[12]=15.0; /*— Protection temps trop cours et grande distance —*/
      }

  /*— 3. On execute le point des que la bank est vide —*/
  do
    {
      ret = robctl(RX90, GET_MOVE_TRAJ_TYPE, (void*)&tmp);
      bk = ROBCTL_RX90_BANK_STATE(tmp);
    }
  while(bk==1);

  /* — 4.6 Mouvement vers le nouveau point — */
```

```
  if(robctl(RX90, PUT_POSITION, (void *)q) == ERROR)
    printf("\nrx90Move : :Probleme robclt position ......");
}


int rx90Pile( double point[LONG_POINT], int config[LONG_CONF], int
mode)
{
  /*— Creation de la Pile de TAILLE_PILE entree —*/
  static double pile[TAILLE_PILE][LONG_POINT];
  static int pile_conf[TAILLE_PILE][LONG_CONF];
  static int haut=0, bas=0; /*— Pointeurs —*/
  static int nbr_elts=0;
  int i;

  if(mode == 0)
    {
      if(nbr_elts == TAILLE_PILE)
        {
          return(0);
        }
      else
        {
          /*— ICI on empile —*/
          for(i=0;i<LONG_POINT;i++)
            pile[haut][i] = point[i];
          for(i=0;i<LONG_CONF;i++)
            pile_conf[haut][i] = config[i];
          haut = (haut+1) % TAILLE_PILE;
          nbr_elts++;
          return(1);
        }
    }
  else
    {
      if(nbr_elts == 0)
        {
          return(0);
        }
      else
        {
          /*— ICI on depile —*/
          for(i=0;i<LONG_POINT;i++)
            point[i] = pile[bas][i];
          for(i=0;i<LONG_POINT;i++)
            config[i] = pile_conf[bas][i];
```

```
            bas = (bas+TAILLE_PILE-1) % TAILLE_PILE;
            nbr_elts--;
            return(1);
          }
      }
}

void rx90Socket()
{
  double point[LONG_POINT];
  int config[LONG_CONF];

  /*-- Ici on recoit la socket et on remplit point --*/

  /*-- Ici on recoit une autre socket et on rempli config --*/

  /*-- Ici on empile --*/
  if(rx90Pile(point, config, 0) == 0)
    printf("\nrx90Socket : :Pile pleine....");

}

void rx90Teleop()
{
  /*-- Variables --*/
  double point[LONG_POINT];
  int config[LONG_CONF];
  double rot[3][3], snaP[3][4], joint[6], joint_vel[6], old_joint[6];
  int ret,traj_type=5,i,j;

  /*
      double joint_rob_vel[6], joint_rob_f[6], old_joint[6],q[13],T;
      int bk,tmp,i,j, BEGIN;
  */

  /*-- On lance le thread qui remplit la pile avec la socket --*/
  rx90Socket(); /*-- A MODIFIER -> THREAD --*/


  /*-- Changement du type de trajectoire --*/
  ret = robctl(RX90, PUT_MOVE_TRAJ_TYPE, (void*)&traj_type);

  /*-- 1. On recupere les premieres donnees x y z A B C dt --*/
  while( rx90Pile(point, config, 0)==0 ) {}

  /*-- 2. On contruit la matrice snaP correspondante --*/
```

```
  rx90RTL_to_sna(point[3], point[4], point[5], rot);
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      snaP[i][j]=rot[i][j];
  snaP[0][3]=point[0];
  snaP[1][3]=point[1];
  snaP[2][3]=point[2];

  /*— 3. Recuperation du premier point BEGIN=1 le reste par defaut —*/
  rx90Commande(snaP, 10.0, 0, 1.0, 1.0, 1, -1, 0, joint_vel, old_joint);

  /*— 4. Sauvegarde de la position courante —*/

  for(;;)
    {
      /*— 1. On depile les donnees x y z A B C dt —*/
      while( rx90Pile(point, config, 0)==0 ) {}

      /*— 2. On contruit la matrice snaP correspondante —*/
      rx90RTL_to_sna(point[3], point[4], point[5], rot);
      for(i=0;i<3;i++)
        for(j=0;j<3;j++)
          snaP[i][j]=rot[i][j];
      snaP[0][3]=point[0];
      snaP[1][3]=point[1];
      snaP[2][3]=point[2];


      /*— 4. Determination des valeurs et vitesses articulaires —*/
      rx90Commande(snaP, point[6], config[0], point[8], point[7], config[1],
config[2], config[3], joint_vel, joint);
      /*— Petite indication pour les etourdis —
        config = [SUSPEND, BEGIN, ABS_POS, ABS_ORI]
        point = [x y z a b c T C_POS C_VEL]
        rx90Commande(snaP, T, SUSPEND , C_Vel, C_Pos, BEGIN,
ABS_POS, ABS_ORI, joint_rob_vel, joint_rob_f)
        ————————————————————————*/

      /*— Point identique au precedent ? —*/
      if(joint[0]!=old_joint[0] || joint[3]!=old_joint[3] ||
         joint[1]!=old_joint[1] || joint[4]!=old_joint[4] ||
         joint[2]!=old_joint[2] || joint[5]!=old_joint[5])
        {
          /*— On bouge vers le nouveau point —*/
          rx90Move(joint, joint_vel, point[6]);
```

```
            /*— On sauvegarde le point atteint —*/
            for(i=0;i<6;i++)
                old_joint[i] = joint[i];
        }
    }
}

void rx90Mouse()
{
  /* — Variables — */
  double snaP[3][4], sna[3][3], rot[3][3], X[6], joint_rob_vel[6],
joint_rob_f[6], q[13];
  double T=1.0, C_Vel = 1.0, C_Pos = 0.5;
  int SUSPEND = 0, BEGIN = 1, ABS_POS = -1, ABS_ORI = 0,i,j,k, Ok
= TRUE, Event;
  char Button;
  int tmp, bk;
  int ret, traj_type=5;

   /*— Nouveau type de trajectoire —*/
   ret = robctl(RX90, PUT_MOVE_TRAJ_TYPE, (void*)&traj_type);

  /* — 1. On initialise snaP a l'identite — */
  for(i=0;i<3;i++)
    for(j=0;j<4;j++)
      snaP[i][j] = (i == j ? 1 : 0);

  /* — 2. Initialisation de la souris — */
  if(xspacemouseInit()==SPACEMOUSE_ERROR)
    exit(0);
  xspacemouseRun();

  /* — 3. Premier point — */
  rx90Commande(snaP, T, SUSPEND, C_Vel, C_Pos, BEGIN , ABS_POS,
ABS_ORI, joint_rob_vel, joint_rob_f);
  BEGIN = 0;

  /* — 4. Attente des evenements — */
  while(Ok)
    {

      if(xspacemouseWait(&Event,&Button,X)==SPACEMOUSE_ERROR)
        Ok=FALSE;
      printf("Mouse data : x=%lf, y=%lf, z=%lf, a=%lf, b=%lf,
c=%lf",X[0],X[1],X[2],X[3],X[4],X[5]);
```

```
/* — 4.1 Determination de la matrice de rotation "rot" — */
rx90RTL_to_sna( X[4]/100000, X[3]/100000, X[5]/100000, rot);
printf("\nVoila la matrice de rotation :");
for(i=0;i<3;i++)
{
  printf("\n");
  for(j=0;j<3;j++)
    printf("%lf ",rot[i][j]);
}

/* — 4.2 Mise a jour du point courant — */
for(i=0;i<3;i++)
  for(j=0;j<3;j++)
    {
      sna[i][j] = 0.0;
      for(k=0;k<3;k++)
        sna[i][j] += snaP[i][k] * rot[k][j];
    }
for(i=0;i<3;i++)
  for(j=0;j<3;j++)
    snaP[i][j] = sna[i][j];
snaP[0][3] -= X[0]/150000;
snaP[1][3] -= X[2]/150000;
snaP[2][3] += X[1]/150000;

/* — 4.3 Affichage — */
rx90Affichemat(snaP);
printf("\nVoila les offsets : dPx=%lf dPy=%lf
dPz=%lf",X[0],X[1],X[2]);

T=0.3;
/* — 4.4 Calcul des positions et vitesses articulaires — */
rx90Commande(snaP, T, SUSPEND, C_Vel, C_Pos, BEGIN ,
ABS_POS, ABS_ORI, joint_rob_vel, joint_rob_f);
for(j=0;j<6;j++)
{
  q[j] = RADTODEG(joint_rob_f[j]);
  q[j+6] = RADTODEG(joint_rob_vel[j]);
}
q[12]=0.3;

/*— Saturation des vitesses a 28 —*/
for(i=6;i<12;i++)
  if(q[i]>28)
  {
    q[i]=28;
```

```
            q[12]=15.0;
          }

        /*— 3. On execute le point des que la bank est vide —*/
        do
          {
            ret = robctl(RX90, GET_MOVE_TRAJ_TYPE, (void*)&tmp);
            bk = ROBCTL_RX90_BANK_STATE(tmp);
          }
        while(bk==1);

        /* — 4.6 Mouvement vers le nouveau point — */
        if(robctl(RX90, PUT_POSITION, (void *)q) == ERROR)
          printf("\nProbleme robclt position ......");
      }
    xspacemouseClose();
}
```

# BIBLIOGRAPHIE


[1] ARA FOCUS GROUP SPRING 1998. Motion capture as a means for data aquisition. http ://vizproto.prism.asu.edu/datacapture/motioncapture1/.

[2] Computer Animation. Types of motion capture systems - full body. http ://sotdev2.tech.purdue.edu/tasg/motion_capture/hardware_general.htm.

[3] F. Courreges. Determination d'une architecture de téléopération bilaterale en presence de retards. http ://www.irisa.fr/manifestations/2000/jjcr/Resumes/51.shtml.

[4] Khalil W. Dombre E. *Modélisation, identification et commande des robots* . Hermès, Paris, 1999.

[5] Pan-Mook Lee Dong-Soo Kwon, Jee-Hwan Ryu and Seok-Won Hong. Design of a teleoperation controller for an underwater manipulator. In *International Conference on Robotics & Automation* , 2000.

[6] G. Burdea et P. Coiffet. *La Réalité Virtuelle*. Hermès, Paris, 1993.

[7] A. Micaelli G. Morel F. Geffard, C. Andriot. On the use of a base force/torque sensor in teleoperation. In *International Conference on Robotics & Automation San Francisco* , 2000.

[8] Ph. D. F.J. Perales Computer Graphics and Vision Group Departement of Computer Science Universitat de les Illes Belears *UIB*. Humain motion analysis & sythesis using computer vision and graphics techniques. state of art and applications. http ://iason.zcu.cz/ skala/IDW/IDW2002/articulo.pdf.

[9] M. Sigut J.L Cruz, B. Cuesta and L. Acosta. Simulation, remote access, and monitoring of a robot in java. In *International Journal of Robotics and Automation, Vol. 17, No. 3*, 2002.

[10] S. Ménardais. Fusion et adaptation temps réel de mouvements acquis pour l'animation d'humanoïdes synthétiques, 2003.

[11] Frank Tendick Murat Cenk Çavuşoğlu, Alana Sherman. Bilateral controller design for telemanipulation in soft environments. In *International Conference on Robotics & Automation*, 2001.

[12] Stelios Gerogiannakis Nikitas M.Sgouros. Integrating wap-based wireless devices in robot teleoperation environments. In *International Conference on Robotics & Automation*, 2002.

[13] Peter Corke Pablo d'Angelo. Using a wap phone as robot interface. In *International Conference on Robotics & Automation* , 2002.

[14] P.J. Sanz R. Marin and A.P. del Pobil. Teleoperated robot system via web : The uji telerobotic training system. In *International Journal of Robotics and Automation, Vol. 17, No. 3*, 2002.

[15] Sharon Schatz. Vicon notion systems' brian nilles : Motion-capture moves forward. http ://www.awn.com/mag/issue4.11/4.11pages/schatzvicon3.php3, 2000.

[16] B. Ghimire K. Vovk G. Gosine T. Sobh, R. Mihali and P. Batra. Web-controlled devices and remote manipulation : Distance learning case studies. In *International Journal of Robotics and Automation, Vol. 17, No. 3* , 2002.

[17] Japan Tohoku University Sendai. Dual-arm space robot teleoperation. http ://www.space.mech.tohoku.ac.jp/research/parm/parm-e.html.