Résumé

Ce mémoire a pour but d'améliorer le projet HandiAccess (simulateur d'accessibilité de l'environnement pour les personnes à mobilité réduite) par l'implémentation de la méthode itérative FABRIK.

Avant de mettre en pratique cette méthode, et pour mieux situer nos travaux, nous avons commencé par présenter les différents outils de modélisation d'un humanoïde et les méthodes de contrôle de mouvement : les méthodes de cinématique inverse et la capture de mouvement. Puis nous avons présenté en détails les méthodes CCD et FABRIK.

Nous sommes passés à l'application de la méthode FABRIK et nous avons comparé ses résultats avec ceux de la méthode CCD. Nous avons constaté que l'application produit des mouvements visuellement lisses, atteignant la position désirée avec un coût en termes de temps de calcul très bas à celles données par la méthode CCD.

Mots-clés : Cinématique inverse, FABRIK, HandiAccess, C++.

Abstract

This thesis aims to improve the project HandiAccess (simulator of environment accessibility for people with reduced mobility) by the implementation of the iterative method FABRIK.

Before implement this method, and to situate our work, we began by presenting the different modeling tools and methods of motion control : methods about inverse kinematics and motion capture. Then we presented in detail the methods FABRIK and CCD.

We passed to the application of FABRIK method and we compared its results with those of the CCD method. We found that the application produces visually smooth movements, reaching the desired position with a low cost in terms of computation time with those given by the CCD method.

Keywords : Inverse kinematics, FABRIK, HandiAccess, C++.

Remerciements

Je remercie Dieu le tout Puissant qui nous a donné la force et la volonté pour réaliser ce modeste travail.

Nous tenons à exprimer notre grande gratitude à notre encadreur Mr MOUSSAOUI, pour avoir accepté de nous encadrer tout au long de ce travail, pour sa disponibilité, son amabilité, ses conseils et suggestions et pour toute l'aide morale qu'il n'a cessé de nous donner.

Nous tenons aussi a remercier notre co-encadreur Mr BOUALEM, pour sa disponibilité, son aide et conseils.

Nous tenons également à remercier Mr MELIANI pour l'honneur qu'il nous fait de présider notre jury de soutenance nous lui exprimons notre gratitude profonde.

Nos remerciements s'adressent ensuite à Mme BENALLAL et Mr BENHABIB qui ont aimablement accepté d'examiner et de juger notre modeste travail.

Sans oublier nos très chères familles et surtout parents pour leur contribution, leur soutien et leur patience.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours soutenus et encouragé au cours de la réalisation de ce mémoire. Merci à tous et à toutes.

Dédicaces

Je dédie ce travail, à tous ceux qui me sont chers,

A ma mère (que Dieu ait son âme), qui a toujours éclairé mon chemin et soutenue tout au long de mes études.

A mon père qui a toujours était un exemple pour moi et qui a fait de moi l'homme que je suis aujourd'hui.

A mes fréres Sido et fawzi pour leur patience, aide et encouragements.

A ma nièce Fatima Zahra qui a toujours fait en sorte que je travaille dans la bonne humeur.

A ma belle-soeur Wassila qui s'est toujours inquiété pour moi et qui a était là dans les moments difficiles.

A Warda qui ne m'as jamais laissé tomber, qui a toujours était derrière moi et pour toute la patience et les encouragements qu'elle m'a donné.

A Ma binôme Imene avec qui j'ai partagé trois années de moments inoubliables et avec qui j'ai eu l'honneur de finir mes études. A toute sa famille KERZABI

A mes amis,

Amine, Imad, pour tous les moments de fou rire en plein cours, ainsi que Othman, Mustapha, Farid, Nadir, Nassim, Narimene...

KARA SLIMANE Djamel

Dédicaces

Je dédie ce travail :

A mes très chers parents qui ont toujours été à mes côtés, qui n'ont jamais cessé de m'encourager et de prier pour moi et qui m'ont donné un magnifique modèle de labeur et de persévérance.

A mes chers frères et sœurs : Amina, Fayçal, Walid et Feriel présents dans tous mes moments d'examens par leurs soutien et encouragements.

A la mémoire de mes grands-parents, A mes proches et à toute La famille KERZABI.

A mes meilleurs amis : Abla, Amine, Zineb, Sara, Imad, G.Samir, Samir qui sont pour moi des frères, sœurs et des amis sur qui je peux compter

A toute la famille KARA SLIMANE.

A tous ceux qui sont proches de mon cœur et dont je n'ai pas cité le nom.

Enfin,

Je tiens à faire parvenir mes sincères reconnaissances à mon meilleur ami djamel dont l'esprit d'équipe et la présence n'ont eu de faille durant tout notre parcours d'étude.

KERZABI Imene Fatima

Table des matières

Table des figures

Liste des tableaux

Abréviations

CCD : **C**yclic **C**oordinate **D**escent.
FABRIK : **F**orward **A**nd **B**ackward **R**eaching **I**nverse **K**inematics.
EDF : **É**lectricité **D**e **F**rance.
CEA : Le **C**ommissariat à l'**E**nergie **A**tomique
INRIA : L'**I**nstitut **N**ational de **R**echerche en **I**nformatique et **A**utomatique
LRV : le **L**aboratoire de **R**obotique de **V**ersailles
LABRI : Le **LA**boratoire **B**ordelais de **R**echerche en **I**nformatique
PSA : **P**eugeot **S**ociété **A**nonyme.
PERF-RV : Plate-forme française de réalité virtuelle
SIG : **S**ystème d'**I**nformation **G**éographique.
HD : **H**aute **D**éfinition.
DDL : **D**egrés **D**e **L**iberté.
DH : **D**enavit-**H**artenberg.
OGRE 3D : Object-Oriented Graphics Rendering Engine.
DLL : **D**ynamic **L**ink **L**ibrary.
XML : e**X**tensible **M**arkup **L**anguage.

Introduction générale

Au cours de ces dernières années, le développement technologique dans le domaine de la réalité virtuelle a connu une avancée importante grâce aux recherches technologiques dans divers domaines, tels que l'automobile, l'aéronautique ou bien encore le cinéma et les jeux vidéo. Le but étant de toujours chercher à améliorer et à optimiser les performances et de réaliser toutes les simulations nécessaires afin d'anticiper et de solutionner les problèmes et éviter les pertes d'argent.

C'est ainsi que de nouvelles voies d'investigation ont été ouvertes comme celle de l'étude de l'environnement avec lequel interagie l'être humain. Elle permet de faciliter et d'améliorer le monde où l'on vit, et plus particulièrement dans le cas des personnes handicapées. Ce genre études nous aident à mieux comprendre les difficultés rencontrées dans le quotidien de cette frange de la population, pour des actions et des gestes qui parraissent anodins mais qui deviennent tellement compliqués car l'espace n'est pas conçu d'après leur point de vue, donc non-adapté à eux. C'est de là qu'ont commencé des recherches sur l'étude de l'accessibilité de l'environnement pour les personnes à mobilité réduite, une sorte d'empathie technologique qui nous aide à mieux nous comprendre et à améliorer le vivre-ensemble.

Dans ce travail nous nous sommes intéressés au projet HandiAccess qui s'inscrit dans la lignée des logiciels qui traitent de l'étude de l'accessibilité de l'environnement pour les personnes à mobilité réduite. Ce grand projet a déjà commencé il y'a deux ans et propose déjà une multitude de fonctionnalités et d'options. Cependant, il ne cesse de s'accroitre et de s'améliorer par des modifications et des ajouts.

Notre contribution à ce projet va être de proposer l'intégration d'une nouvelle méthode itérative : la méthode FABRIK. Notre choix a été motivé par les nombreux avantages que présente cette méthode a commencé par sa simplicité et d'autres qui seront développé dans notre mémoire. On pourra aussi comparer et évaluer les résultats obtenus par cette méthode avec les résultats d'une méthode déjà existante et qui est la CCD.

Ainsi notre mémoire s'articule autour de trois chapitres :

Le chapitre 1 est une introduction à la robotique, il présente aussi une description de l'anatomie humaine afin de faire l'analogie entre le corps humain et ses modèles physiques.

Le chapitre 2 est une présentation détaillée des deux méthodes CDD et FABRIK.

Le chapitre 3 contiendra la présentation du logiciel HandiAccess, l'implémentation de

la méthode ainsi que l'analyse et l'étude des résultats obtenus.

Nous conclurons ce mémoire par un bilan et des perspectives pour une amélioration future.

Motivation

Les personnes en situation de handicap sont victimes d'une double peine : leur handicap, mal pris en charge généralement et les difficultés de vie dans une société qui ne leur est pas adaptée.

Selon les estimations des Nations Unies, dans tous les pays les personnes handicapées représentent 7 à 10 pour cent de la population[1], mais il est évident que ce pourcentage peut être plus fort. L'estimation ci-dessus indique clairement que cette tranche de la société ne constitue pas un groupe minoritaire marginal, particulièrement si l'on considère que le handicap qui frappe une personne n'affecte pas uniquement sa propre situation mais également celle de sa famille, voire de sa communauté.

Donc le problème concernant les personnes handicapées est considérable, et mérite une attention particulière, car il ne faut surtout jamais oublier que nul n'est à l'abri d'un handicap.

En Algérie, La situation des personnes handicapées est dégradante et de plus en plus détériorée. Ils rencontrent de gros problèmes de mobilité et de déplacements : ils éprouvent de grandes difficultés pour circuler sur les trottoirs, monter des escaliers, utiliser les transports en commun ou encore avoir accès aux habitations ou aux services publics. De plus, Il n'existe pas de réglementation en matière d'accessibilité des lieux publics, transports ou autre, qui entre dans la synergie de faciliter la mobilité de cette frange de population qui prend de plus en plus d'ampleur. Et c'est ce qui nous a motivés le plus à travailler sur ce projet et essayer d'apporter des améliorations afin qu'il soit plus performant et qu'il puisse être apte à bien fonctionner.

FIGURE 0.0.1: Lieu non accessible

Travaux antécédents

La réalité virtuelle n'est pas née spontanément. Comme toute nouvelle technique, elle a eu des antécédents qui ne s'appelaient pas encore «réalité virtuelle». Principalement, les simulateurs de transport ont permis à des professionnels d'interagir avec un environnement partiellement virtuel ; depuis 50 ans environ, on faisait de la réalité virtuelle sans le savoir [2].

L'équipe RV [1] a acquis des compétences dans le domaine de la réalité virtuelle depuis 1992, grâce à des projets de recherche portés par un ensemble de partenaires représentant des acteurs du secteur industriel et des laboratoires de recherche comme le CEA, INRIA, EDF, LRV, Renault, PSA Peugeot Citroën, LABRI, etc. [3]

Dans cette partie on en citera quelque-uns :

Projets : AccesSim, Handisim et AccesSig

Les trois projets AccesSim, Handisim et AccesSig visent au développement de systèmes de Réalité Virtuelle pour mettre en situation des utilisateurs à mobilité réduite afin de vérifier leurs accessibilités.

Projet AccesSim

Le projet AccesSim est un projet soutenu par la région française Ile de France. Il vise au développement d'un système de Réalité Virtuelle constituant un outil innovant d'aide à la conception d'environnements accessibles [4].

1. Équipe qui travaille dans le projet PERF-RV

FIGURE 0.0.2: Interface de AccesSim.

Projet Handisim

Le fauteuil roulant virtuel Handisim a été créé par des étudiants en informatique de Laval en Mayenne (France). [5].

Il vise à sensibiliser la population et les architectes urbains aux difficultés de déplacements et de mobilité que rencontrent les utilisateurs de fauteuil roulant dans les villes.

Donc, c'est au grand public que s'adresse ce projet. Chacun pourra, grâce à cette réalisation, se rendre compte de façon très réaliste des obstacles rencontrés quotidiennement par les personnes handicapées moteur.

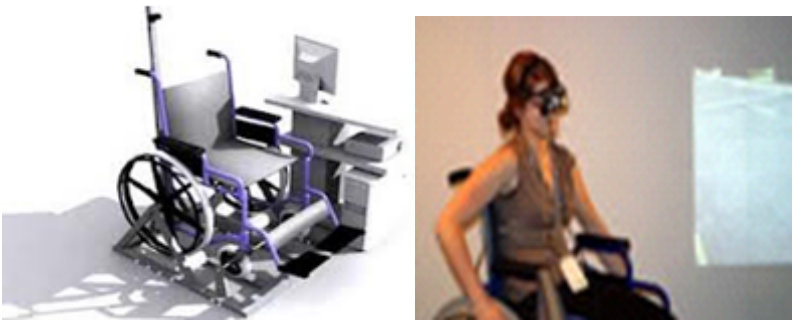


FIGURE 0.0.3: Fauteuil roulant virtuelle Handisim

Projet AccesSig

Le projet AccesSig, en partenariat avec la communauté d'agglomération de Saint-Quentin aux Yvelines (France), vise à développer un calculateur d'itinéraires qui prendra

en compte les besoins spécifiques des personnes à mobilité réduite pour leur assurer un cheminement accessible. Aussi, il mettra en place une plate-forme d'échange SIG (Service d'Information Géographique) pour améliorer les conditions d'accessibilité de la voirie et des espaces publics [4].

Projet Handiman

Ce projet crée une base de données de mouvements d'accessibilité permettant de simuler n'importe quel individu dans n'importe quel véhicule sans maquette physique. Son objectif est de simuler les mouvements (ou stratégies) d'entrée/sortie des véhicules des personnes âgées ou handicapées, et de quantifier l'inconfort lors de ces mouvements [6].

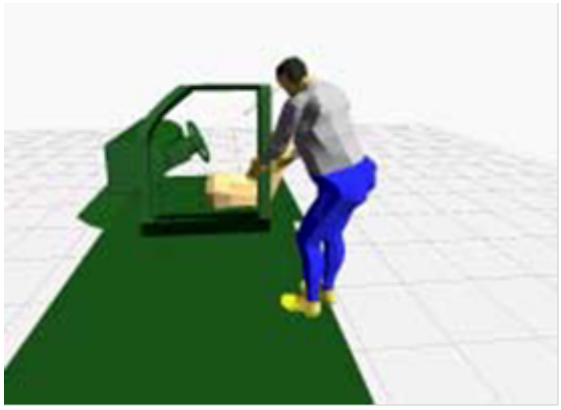


FIGURE 0.0.4: Projet Handiman

Projet PSA Peugeot

L'intérêt de PSA Peugeot Citroën pour la réalité virtuelle a démarré en 1999. De 1999 à 2005, le groupe a investi plus de huit millions d'euros dans les systèmes immersifs.

En collaboration avec Scalable Graphics et NVIDIA, PSA a mis au point une installation pour son centre de réalité virtuelle basée sur la technologie Move. Le tout est conçu pour créer l'illusion d'une immersion totale dans un monde virtuel, elle contient un écran à l'échelle 1/1 doté d'une stéréo passive et d'un holobench en stéréo active en forme de L.

FIGURE 0.0.5: Move en L

L'établissement bénéficie de solides performances, avec des rendus jusqu'à 400 images Full HD par seconde, facilement atteints tout en étant conformes aux résolutions 4K et au-dessus. Enfin, il repose sur des composants standards pouvant être améliorés par la suite pour des simulations de réalité virtuelle encore plus contraignantes [7].

FIGURE 0.0.6: Projet PSA Peugeot

Projet Renault

Pour répondre aux besoins du marché et à la concurrence, Renault diminue son cycle d'étude des projets véhicule tout en intégrant des innovations. Le Centre Technique

de Simulation Renault a construit et exploité cinq (05) simulateurs de conduite qui sont utilisés comme moyens d'essais virtuels dans différents domaines de l'ingénierie automobile : ergonomie, conception des éclairages, prestations dynamiques, systèmes d'aide à la conduite, étude du comportement du conducteur, etc.[8]

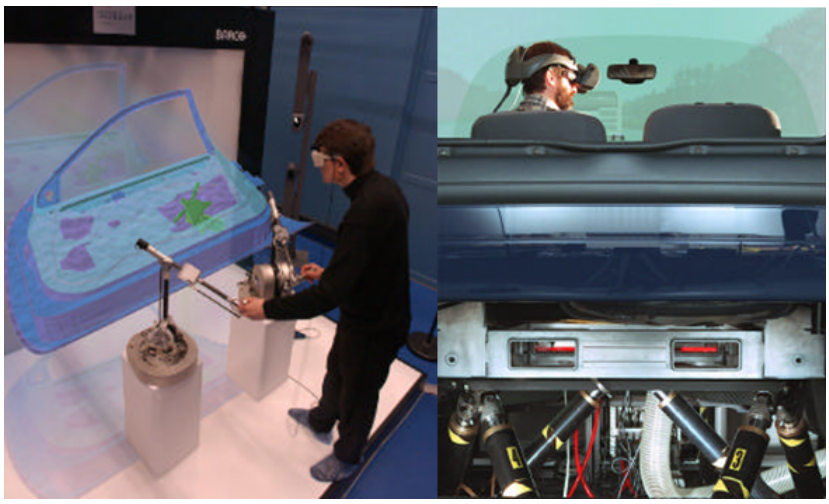


FIGURE 0.0.7: Projet Renault

Chapitre 1

Les bases de la robotique

1.1 Introduction

Dans le cadre d'entités virtuelles, en particulier lors de la représentation et la simulation d'humains, le réalisme, la crédibilité, la rapidité de calcul, et la facilité de contrôle des mouvements sont généralement les éléments primordiaux.

Pour simuler des humanoïdes virtuels, il est nécessaire de définir un modèle les représentant.

Les nouvelles tendances de la robotique laissent à entrevoir une place grandissante des robots humanoïdes dans la vie courante des humains dans des domaines comme l'aide aux personnes âgées, l'éducation, la médecine...

Pour cela, nous nous intéressons dans ce chapitre à donner un rappel sur la composition de quelques parties du corps humain et de voir les notions de base de la robotique qui nous seront utiles par la suite.

1.2 Modélisation de la structure squelettique

1. Le squelette : un squelette est défini par un ensemble de chaines représentant le corps d'un objet (dans notre cas, le corps humain). Un mouvement appliqué au squelette entraîne la déformation de son enveloppe.
2. Une enveloppe : une enveloppe défini l'extérieur visible de l'objet [9].
3. Une chaine : une chaine est composée de :

– Une racine : début de la chaine articulée.
– Un joint : point de rotation qui relie deux segments.
– Une liaison : segment entre deux joints.
– L'effecteur de fin (point de contrôle) : placé au bout de la chaine, il permet d'invoquer la cinématique inverse [9].

1.2.1 Les articulations

Egalement appelées jointures, les articulations correspondent au mode d'union et de connexion des os les uns avec les autres. Les articulations peuvent être extrêmement variée suivant le type de mouvement relatif qu'elles autorisent et peuvent impliquer plusieurs degrés de liberté[10].

Le tableau (1.1) si dessous illustre les différents types d'articulation [10].

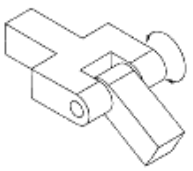
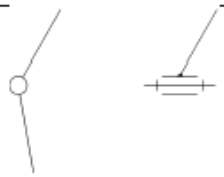
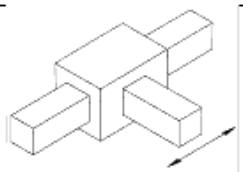
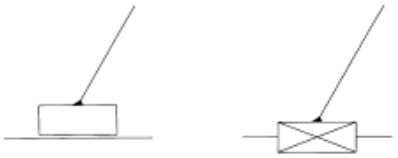
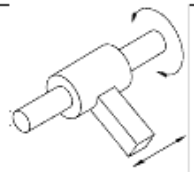
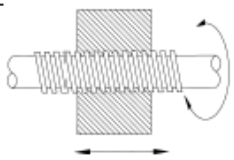
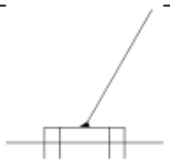
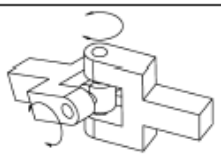
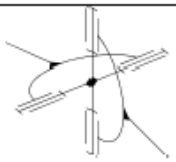
Type d' articulation	*Forme physique*	*Symbol*	*Degrés de liberté*
Rotation (R)			1
Prismatique (P)			1
Cylindrique (C)			2
"Screw" (S)			1
Sphérique (G)			3
Universelle (U)			2

TABLE 1.1: Principaux types d'articulations et leur représentation conventionnelle

1.2.2 Structure osseuse du membre supérieur

L'étude biomécanique du mouvement humain s'appuie sur une description anatomique des membres impliqués dans le geste ou la posture analysée. Dans notre cas, on s'intéresse uniquement à la partie supérieure du corps humain.

Alors, on présentera dans ce qui suit les différentes zones corporelles ainsi que leurs mouvements principaux afin de pouvoir proposer une modélisation adaptée à la mise en œuvre d'une méthode d'animation de la partie haute du corps humain.

1.2.2.1 L'épaule

Le complexe articulaire de l'épaule est composé de trois (03) os : l'omoplate, la clavicule et l'humérus. [11].

C'est l'articulation la plus mobile de l'organisme, Elle est essentiellement composée de deux articulations :

- L'articulation principale entre omoplate et humérus (3 degrés de liberté).
- L'articulation accessoire entre omoplate et clavicule (3 degrés de liberté).

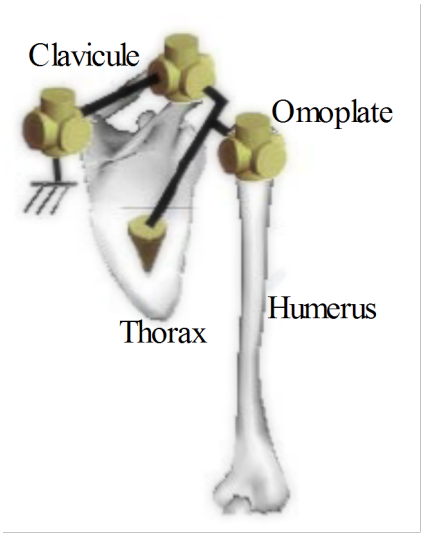


FIGURE 1.2.1: Anatomie et modélisastion de l'épaule [12].

1.2.2.2 La colonne vertébrale

La colonne vertébrale est véritablement notre arbre de vie. Elle se compose de vertèbres bien évidemment mais également de disques intervertébraux situés entre les vertèbres, ils confèrent au rachis sa mobilité[13].

1. La colonne vertébrale est formée d'un assemblage de 32 à 34 os.
2. Elle présente cinq régions :

- La région cervicale : 7 vertèbres.
- La région dorsale : 12 vertèbres sur lesquelles s'articulent les côtes .
- la région lombaire : 5 vertèbres.
- Le sacrum : 5 vertèbres sacrées, soudées les unes aux autres et reliées au bassin.
- Le coccyx : 3 à 5 vertèbres coccygiennes.

Un modèle de douze degrés de liberté a été développé pour exprimer ces parties, il est capable d'exprimer les différents gestes possibles de la colonne, comme montré dans la figure (1.2.2) suivante :

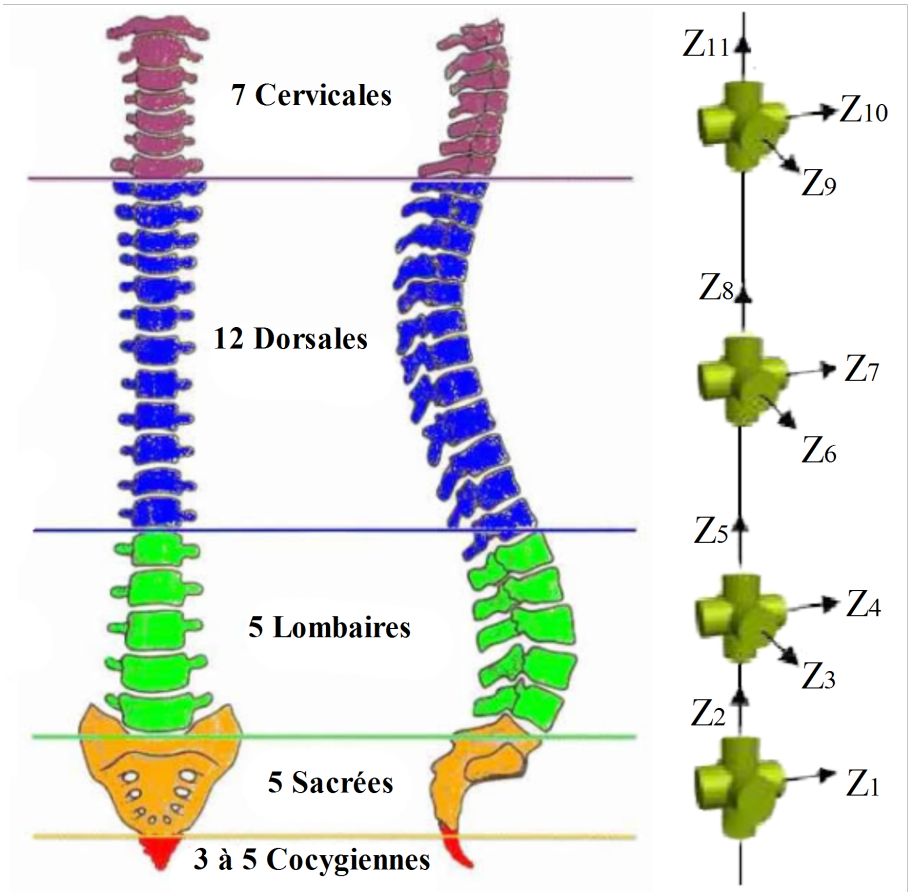


FIGURE 1.2.2: Colonne vertébrale [14].

1.2.2.3 Le bras

La figure (1.2.3) suivante décrit la structure du bras

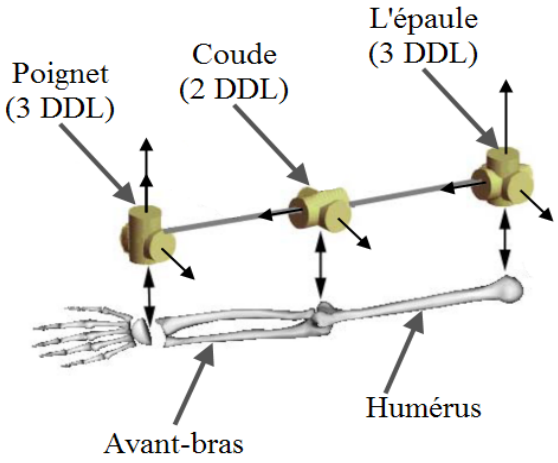


FIGURE 1.2.3: Structure du bras.

Le bras contient deux articulations :
- le coude : est la partie qui relie l'humérus à l'avant-bras (2 degrés de liberté).
- le poignet : est la partie qui relie l'avant-bras à la main (3 degrés de liberté).

La main

La main est composée de 27 os que l'on divise en trois parties [15] :
- 8 os du carpe : constituant le poignet et qui sont disposés en 2 rangées.
- 5 os métacarpiens : qui constituent la paume de la main.
- 14 phalanges : 3 par doigt et 2 pour le pouce.

Elle est généralement représentée par un modèle de 26 degrès de liberté.

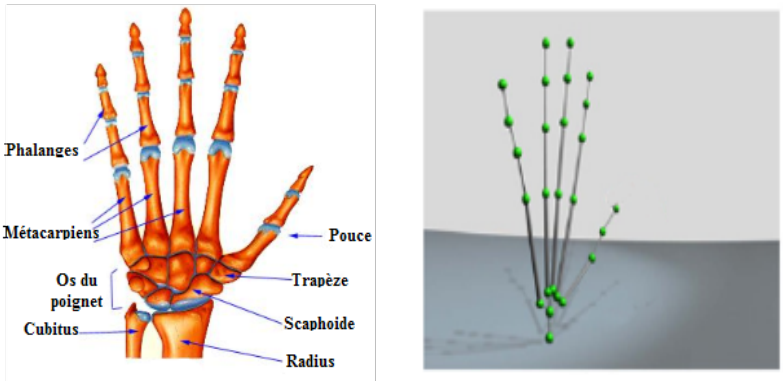


FIGURE 1.2.4: Anatomie et structure de la main[15, 16].

1.3 Outils de modélisation

1.3.1 Paramètres de Denavit-Hartenberg *(DH) :*

C'est l'une des méthodes les plus utilisées pour le calcul des modéles géométriques lorsqu'il s'agit d'une chaine d'articulations simple et ouverte.

Pour détérminer ces paramètres, il faut suivre les étapes suivantes [17] :

– Donner un repère pour chaque corps (articulation).
– Les axes Z_i s'allignent suivant les axes des liaisons.
– Les axes X_i s'allignent pour être perpendiculaires à $Zi-1$ et Zi, et $X_i = Z_i \wedge Z_{i+1}$.
– Les axes Y_i sont choisis de manière à former un trièdre direct avec les axes Z_i et X_i.

Compte tenu de tout ce qui précède textuellement, on peut définir les paramètres DH comme suit :

a_{i-1} : distance entre z_i et z_{i-1}, mesurée le long de x_{i-1}
α_{i-1} : angle entre z_{i-1} et z_i, mesuré autour de x_{i-1}
d_i : distance entre x_{i-1} et x_i, mesurée le long de z_i
θ_i : angle entre x_{i-1} et x_i, mesuré autour de z_i
La figure (1.3.1) suivante illustre les descriptions ci-dessus [18] :

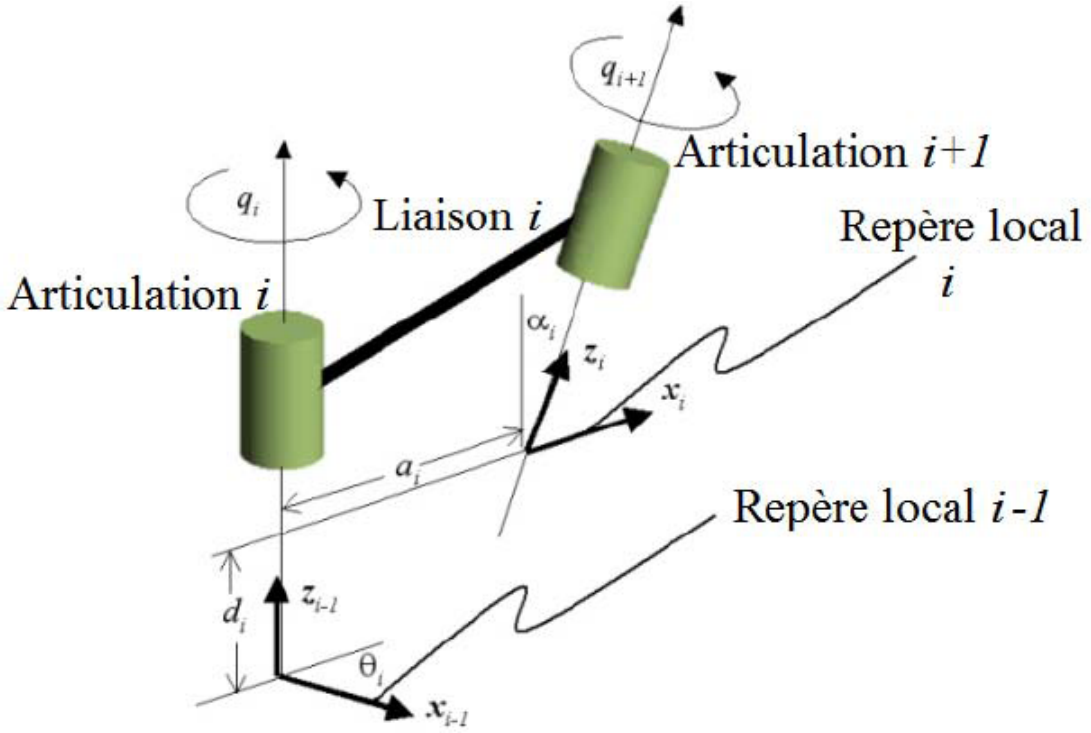


FIGURE 1.3.1: Représentation de Denavit et Hartenberg

Matrice de transformation homogène d'un lien

Pour des raisons de simplicité, on utilise une matrice homogène qui permet de calculer un changement de repère en une seule opération matricielle[19].

C'est de R_{i-1} vers R_i. Elle est décomposée en quatre (04) transformations élémentaires comme déja vu ci-dessus :

1- Rotation autour de z d'un angle θ_i.

2- Translation le long de z d'une distance d_i.

3- Translation le long de x d'une distance a_i.

4- Rotation autour de x d'un angle α_i.

La matrice de transformation homogène s'écrit donc :

$$ {}^{(i-1)}T_i = \begin{bmatrix} cos\,\theta_i & -cos\,\alpha_i\,sin\,\theta_i & sin\,\alpha_i\,sin\,\theta_i & a_i\,cos\,\theta_i \\ sin\,\theta_i & cos\,\alpha_i\,sin\,\theta_i & -sin\,\alpha_i\,cos\,\theta_i & a_i\,sin\,\theta_i \\ 0 & sin\,\alpha_i & cos\,\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

1.3.2 Les équations de Lagrange

Pour les mécanismes robotiques, les équations du mouvement peuvent être obtenues en utilisant le formalisme de Lagrange ou d'autres méthodes.

Le formalisme des équations de Lagrange va nous permettre de déterminer la structure générale du modèle dynamique du robot caractérisé par des couples pour le cas des articulations rotatives et par des forces pour les articulations prismatiques.

Ces équations du mouvement sont définies par le Lagrangien $L(q, \dot{q})$ qui est la différence explicite de l'énergie cinétique T et de l'énergie potentielle U.

Le lagrangien L est donné par la forme suivante [20] :

$$ L(q, \dot{q}) = T(q, \dot{q}) - U(q) \qquad (1.1) $$

L'équation de Lagrange du mouvement sans contraintes est donnée par l'expression ci-dessous :

$$ \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \qquad (1.2) $$

Où :

$i = 1, 2, \ldots\ldots n$.

q_i est la position de la $i$ème coordonnée généralisée du système.

τ_i est la force de la $i$ème coordonnée généralisée du système.

1.4 Méthode de contrôle de mouvements géométriques

Il s'agit des méthodes qui privilégient les informations de type géométrique (position et orientation des liens essentiellement dans le cas de structures articulées) [21].

On trouve dans les méthodes géométriques :

1.4.1 Animation par capture de mouvement *(Motion capture)*

Appelée aussi en terme anglais "motion capture", c'est un procédé d'animation d'entité virtuelle généralement assimilé au mouvement humain.

Elle consiste à capter un mouvement réel puis l'appliquer à un univers virtuel. À l'heure actuelle c'est le procédé le plus efficace en matière de qualité, de fiabilité et de précision.

On distingue deux types de matériel de capture de mouvement :

1. Les systèmes basés sur des cellules photosensibles et marqueurs actifs

   C'est une méthode algorithmique qui consiste à identifier les informations reçues par les capteurs disposés sur le corps de l'être humain. Ces informations vont être traduites par la suite à l'aide d'algorithmes bien spécifiques. Le principal inconvénient de cette technique c'est qu'elle nécessite de porter un matériel électronique comme un câble encombrant les mouvements de la personne ou une batterie et une connexion sans fil limitant la durée maximum de capture.

2. Les systèmes basés sur des caméras et marqueurs passifs réfléchissants

   Ici les marqueurs sont aussi placés sur l'ensemble du corps humain, ce dernier est entouré de caméras qui filment le personnage en différents points de vue, pour cela les marqueurs doivent être bien placés afin d'être souvent visible.

   Enfin on pourra construire une image en 3D à partir de tous les enregistrements. Mais c'est une méthode qui nécessite la disponibilité de suffisamment de matériel pour pouvoir suivre tous les marqueurs.

L'utilisation de ces systèmes est très intéressante vue ses multiples avantages mais n'empêche qu'elle reste une méthode qui est dans la plupart des cas très chère [22].

1.4.2 Cinématique inverse

La cinématique inverse est un procédé d'animation et d'articulation qui permet de trouver les valeurs des angles d'une chaine cinématique afin de placer cette chaine dans une certaine posture, répondant à un certain nombre de contraintes. Mathématiquement, le problème de la géométrique inverse est régi par l'équation suivante :

$$q = f^{-1}(X) \qquad (1.3)$$

Où $f^{-1}(X)$ représente un système de n équations non-linéaires avec n inconnus.

1.4.2.1 Méthodes de résolutions de la cinématique inverse

Méthodes symboliques ou analytiques

Ce sont des méthodes qui donnent des résultats algébriques minimaux. Elles se basent sur le calcul mathématique direct pour la résolution des équations qui modélisent le système, en prenant en compte le maximum de ses propriétés.

Ce genre de méthode reste efficace pour des cas simples et faciles à résoudre mais elles ont l'inconvénient d'être sensible au nombre d'inconnus, aux degrés des équations et aussi à ne pas prendre en compte certaines contraintes [22].

Méthodes adaptatives

Ces méthodes de résolution proviennent d'autres domaines d'applications, et ont été testées sur des systèmes robotiques, nous citerons entre autres celles basées sur le raisonnement géométrique, les réseaux de neurones [23], ou encore les algorithmes génétiques [24].

Ces méthodes peuvent s'avérer très efficaces, car ayant été testées sur certains problèmes en robotique qui ont été résolus avec succès, elles se restreignent tout de même à des problèmes spécifiques seulement [25].

Méthodes itératives

Ce sont des méthodes algorithmiques utilisées pour résoudre un problème d'optimisation. Elles sont considérées comme des procédés itératifs, puisqu'en commençant par une valeur initiale, On converge à la solution optimale après une succession d'itérations et chaque itération nous donne une solution approximative. Certes, auparavant, ces méthodes étaient lentes à s'exécuter, puisqu'elles s'opéraient en des temps de calcul importants, mais vue l'augmentation de la puissance des ordinateurs elles deviennent très recommandées [23, 25].

a.Méthodes basées sur la jacobienne

Les méthodes les plus courantes en termes de cinématique inverse font intervenir la matrice Jacobienne J du système.

La matrice Jacobienne représente les dérivées premières du système pour chaque degré de liberté, c'est-à-dire la direction dans laquelle vont évoluer les effecteurs si on applique une légère variation aux valeurs des degrés de liberté.

La matrice Jacobienne inverse permet une réciproque qui nous permet de déterminer dans quelle direction faire varier les angles de degrés de liberté pour que l'effecteur se déplace dans la direction voulue (la cible).

Malheureusement le calcul de l'inverse d'une matrice quelconque est un problème n'ayant pas de solution dans beaucoup de cas concernant la cinématique inverse. On

utilise par conséquent d'autres méthodes donnant des solutions approchées mais acceptables. Ces solutions sont nombreuses, on peut citer :

1. La Jacobienne transposée
   Consiste à remplacer la matrice inverse de J par sa transposée ce qui donne
   $$\triangle q = \alpha J^T \triangle x \qquad (1.4)$$
   Où
   $\triangle x$: le vecteur de distance entre la position actuelle de l'objet final et la position désirée.
   α : un scalaire
2. La pseudo-inverse
   La notion de pseudo-inverse est utilisée en mathématique pour généraliser la notion de matrice inverse dans des cas non inversibles.
   elle est calculé comme suit :
   $$J^+ = \lim_{\delta \to 0}(J^T J + \delta I)^{-1} J^T = \lim_{\delta \to 0} J^T (J^T J + \delta I)^{-1} \qquad (1.5)$$

b.Méthode basée sur le gradient

– Descente de gradient

C'est une méthode d'optimisation considérée comme étant la plus facile et la plus simple. Elle permet de converger vers la solution qui est le minimum local après un certain nombre d'itérations et en commençant par une valeur aléatoire. Tout cela se fait en suivant l'algorithme suivant :

1. Choisir une valeur initiale x_0 et un seuil $\varepsilon \geqslant 0$.
2. Définir une suite $x1, x2, ..., x_k, x_{k+1}$ par itération en suivant les étapes suivantes :
   a) Calcul du gradiant noté $\nabla f(x_k)$
   b) Test d'arrêt : si $\| \nabla f(x_k) \| \leqslant \varepsilon$, arrêt.
      sinon
   c) Calcul de $\alpha_k > 0$
   d) Nouvel itéré $x_{k+1} = x_k - \alpha_k . \nabla f(x_k)$

c.Méthode Monte-Carlo

C'est une méthode d'approximation connue par l'utilisation du hasard pour résoudre les problèmes de calculs. Elle est basée sur l'idée de prélever des solutions aléatoires sous forme d'échantillons. Cet échantillonnage va être déplacé suivant le mouvement (qui est à son tour aléatoire) afin de trouver l'optimum. Alors plus le tirage aléatoire des solutions est grand et l'algorithme à plus de chances de converger vers la bonne solution et c'est ce qui peut alourdir le calcul [26].

d.Méthode Descente de Coordonnées Cycliques CCD

L'algorithme Cyclic Coordinate Descent (CCD) [27], propose une méthode itérative pour amener l'effecteur sur la cible. Il se base sur l'alignement successif des segments avec la cible. Au fur et à mesure des itérations, l'effecteur va converger vers la cible. Le principal inconvénient de cette méthode est que de fortes discontinuités sont provoquées sur la posture. Cependant elle est moins coûteuse en calculs et surtout n'est pas sensible aux singularités.

e.Méthode FABRIK (Forward And Backward Reaching Inverse Kinematics)

Propose un nouvel algorithme de cinématique inverse qui permet en parcourant la chaîne en avant et en arrière en un nombre fini d'itérations d'atteindre une cible. L'intérêt principal de cette approche réside dans la possibilité de contrôler des systèmes multi-effecteurs mais également dans sa rapidité d'exécution.

1.5 Conclusion

On a vu dans ce chapitre les principales méthodes de modélisation des mouvements spécialement celles utilisant le principe de la cinématique inverse et on a pu voir leurs principales caractéristiques.

Dans le chapitre suivant, on va s'intéresser aux deux méthodes Descente de Coordonnées Cycliques (CCD) et FABRIK.

Chapitre 2

FABRIK et CCD

2.1 Introduction

La production de poses et de mouvements réaliste c'est révélée être un véritable challenge dans les domaines de la robotique et de l'animation de synthèse, ainsi on a vu le développement d'une multitude de techniques servants à résoudre les différents problèmes de la cinématique inverse.

L'une des méthodes la plus connue et la plus utilisée est la Descente de Coordonnées Cycliques *(Cyclic Coordinate Descent)* (CCD). Cette méthode a été massivement utilisée dans l'industrie du jeu vidéo et dans d'autres domaines, due à ses avantages multiples (facilité et rapidité de calcul, poses réaliste, prise en compte des contraintes...).

FABRIK*(Forward and Backward Reaching Inverse Kinematic)* est une nouvelle méthode itérative qui présente au premier abord autant de qualités que la méthode CCD.

C'est pour ça que dans ce chapitre, on va étudier ces deux méthodes de façon plus détaillée pour pouvoir les comparer par la suite.

2.2 CCD

C'est une méthode itérative qui a comme avantage de limiter les calculs (pas de jacobienne ni d'inversion) et d'optimiser globalement le mouvement du système.

2.2.1 Principe de la méthode

La CCD cherche à minimiser l'erreur commise sur la position de l'effecteur final en ajustant l'angle de chaque joint un par un, séquentiellement, cherchant la position permettant de rapprocher au maximum chaque effecteur de sa cible en partant de l'extrémité vers la racine.

Le segment le plus éloigné de la chaîne est orienté de manière à aligner sa racine, l'extrémité de la chaîne et la Cible à atteindre.

Le second segment est ensuite orienté de manière à aligner à son tour sa racine avec la cible.

Les orientations des autres segments sont calculées de la même manière. Une fois parvenu au segment racine, la séquence d'opérations est recommencée à partir de l'extrémité de la chaîne, jusqu'à convergence [28].

Cependant, les algorithmes de type CCD peuvent nécessiter un très grand nombre d'itérations avant d'obtenir un résultat comme ils tendent à favoriser les joints finaux. Ceci peut produire des mouvements à l'apparence non naturelle [21].

Cette méthode est particulièrement indiquée pour la résolution de problèmes ne disposant que d'un seul effecteur mais marche mal dans le cas d'une résolution multiple.

La figure (2.1.1) ci-dessous résume les étapes de la méthode CCD [28] :

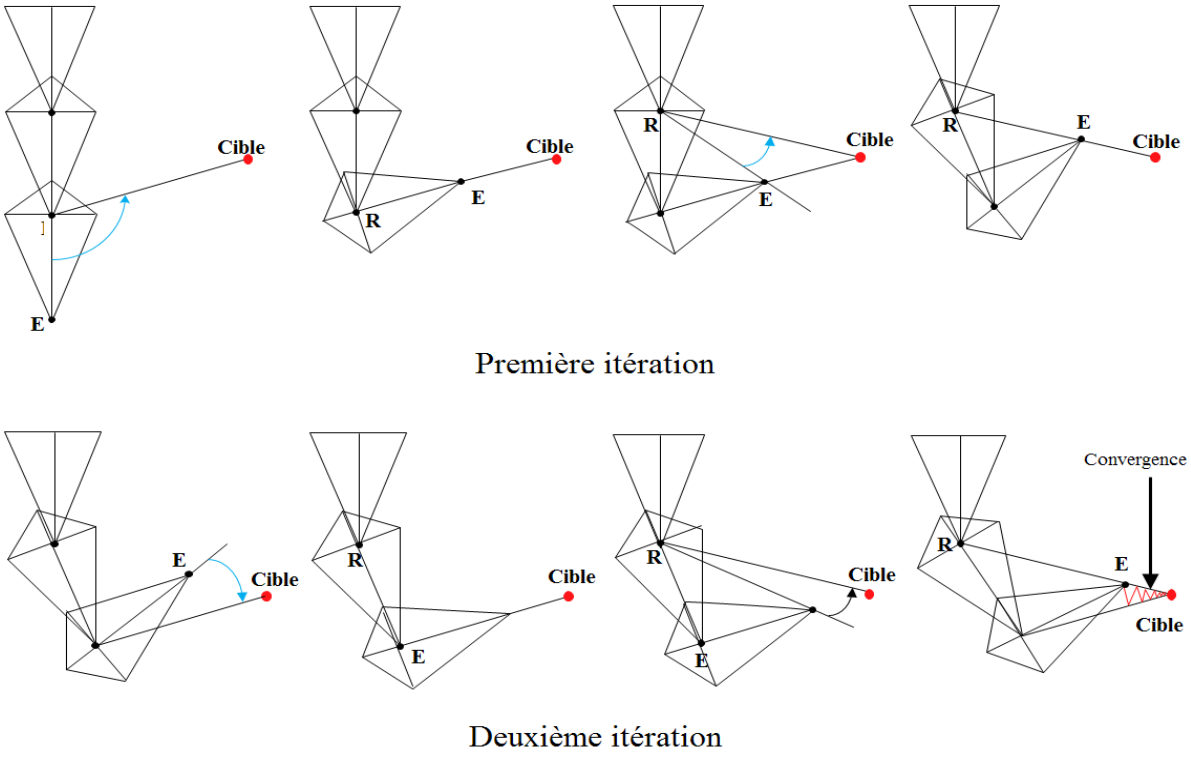


FIGURE 2.2.1: Etapes de la méthode CDD

2.2.2 Cas de CCD

Prenons les données suivantes :

$P_E(q) = [X_E\, Y_E\, Z_E]$: Position de l'extrémité (Point final ou effecteur de fin) en fonction de q.

$[R_E(q)] = [E_1(q)\, E_2(q)\, E_3(q)]$: orientation de l'extrémité en fonction de q. avec $E_i (i = 1, ..., 3)$ les vecteurs unités le long des axes X_E, Y_E *et* Z_E

$P_d = [X_d\, Y_d\, Z_d]$: position désirée de l'extrémité.
$R_d = [d_1\, d_2\, d_3]$: orientation désirée de l'extrémité.
avec $d_j (j = 1, ..., 3)$ sont les vecteurs unités le long des axes $X_d\, Y_d$ et Z_d
$q = [q_1, q_2, q_3]^T$ est le vecteur des variables d'articulations.
$\underline{q_i}$ limite inférieure sur l'articulation q_i
$\overline{q_i}$ limite supérieure sur l'articulation q_i

2.2.2.1 Problème 1 : cinématique inverse avec contraintes

Le probléme 1 est de trouver le vecteur des articulations q tel que :
$P_E(q) = P_d$
$R_E(q) = R_d$
et $\underline{q_i} \leqslant q_i \leqslant \overline{q_i}$

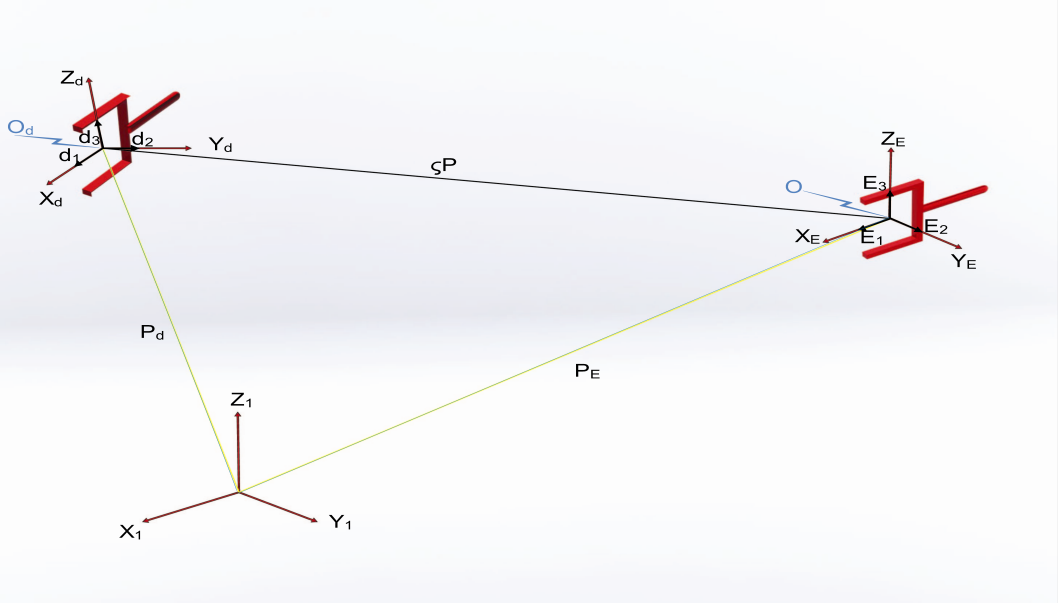


FIGURE 2.2.2: La configuration courante et désiré de l'objet final[19]

Afin de transformer le problème 1 en un problème d'optimisation, on doit définir une fonction appelée la fonction objective notée ci-dessous :

$$E(q) = \Delta P(q) + \Delta O(q) \qquad (2.1)$$

L'objectif est de minimiser cette fonction, sachant que l'erreur de position est donnée par :

$$\Delta P(q) = (P_E(q) - P_d)^T (P_E(q) - P_d) \qquad (2.2)$$

et l'erreur de l'orientation est donnée par :

$$\Delta O(q) = \sum_{j=1}^{3} \sigma_j (d_j.E_j(q) - 1)^2 \qquad (2.3)$$

$$\Rightarrow \Delta O(q) = \sigma_1 (d_1.E_1(q) - 1)^2 + \sigma_2 (d_2.E_2(q) - 1)^2 + \sigma_3 (d_3.E_3(q) - 1)^2 \qquad (2.4)$$

avec σ_1,σ_2 etσ_3 des pondérations logiques.

2.2.2.2 Problème 2 : Cinématique-Minimsation

Si q n'est pas une solution du problème 1 alors on cherchera une nouvelle solution $q^\star$ prêt de la solution optimale.

D'une façon formelle, il faut trouver $q^\star$ tel que : $Er(q^\star) \leqslant \varepsilon$

$Er(q^\star)$: erreur de position et d'orientation.

avec $\underline{q_i} \leq q_i \leq \overline{q_i}$ $\qquad i = 1, ..., n$

ou $\varepsilon > 0$ est la tolérance qui devrait être presque nulle.

La méthode CCD s'exécute en plusieurs cycles, chaque cycle consiste à minimiser l'erreur par rapport à q_n puis à q_{n-1} jusqu'à q_1.

- Minimisation par rapport à q_i :

Deux possibilités existent :

– L'articulation i est rotoide avec la variable d'articulation θ_i.

– L'articulation i est prismatique avec la variable d'articulation S_i

Comme montrer dans la figure (2.2.2) suivante [27] :

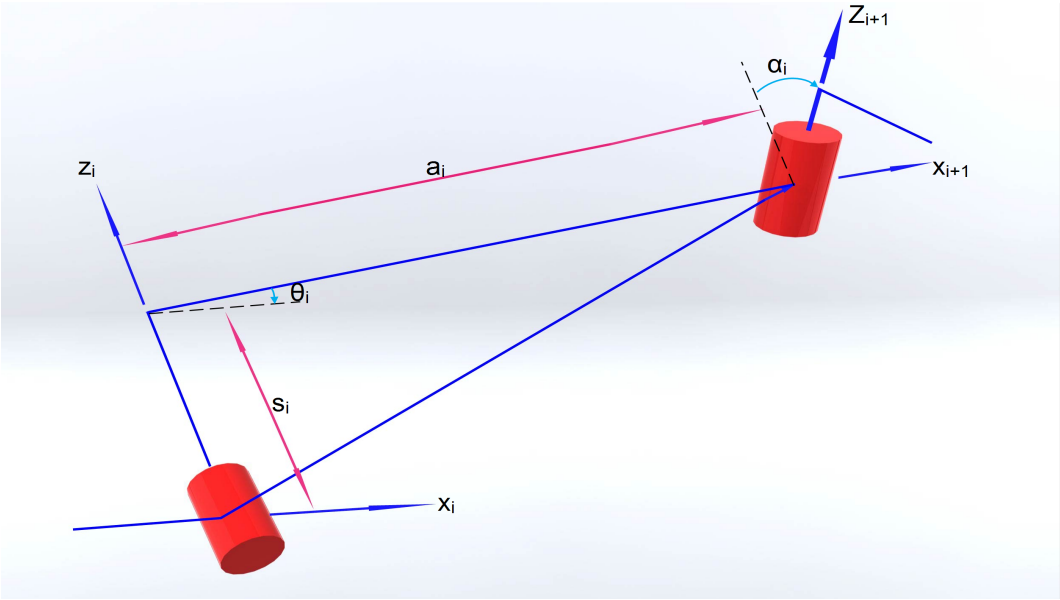


FIGURE 2.2.3: Définition des paramètres des articulations

Cas d'une articulation rotationnelle

Ici, la variable d'articulation q_i est l'angle θ_i, alors la position P_E sera pivotée autour de l'axe Z_i sans toucher aux autres articulations c'est-à-dire $P_1, ..., P_i$ et P_d resteront constantes.

Donc la nouvelle position de l'extrémité devient :

$$P'_{iE}(\phi) = [R(Z_i, \phi)]P_{iE} \qquad (2.5)$$

P_{iE} : représente la position actuelle du point P_i.
O_i : est l'origine du $i$ème système de coordonnées .
P'_{iE} : est le vecteur allant de O_i vers la position actuelle de l'objet final .
La figure (2.2.3) suivante illustre ces paramètres [27] :

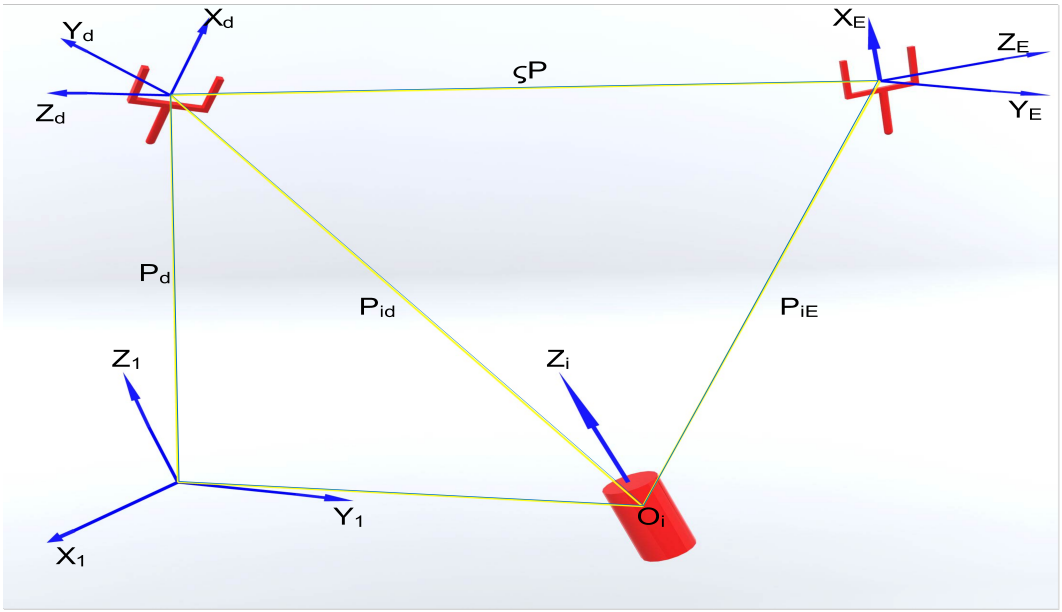


FIGURE 2.2.4: Représentation des paramètres pour une étape de la CCD

L'erreur de position devient
$\Delta P(\phi) = (P_{id} - P'_{iE}(\phi))^T(P_{id} - P'_{iE}(\phi)) \qquad (2.6)$
$\Rightarrow \Delta P(\phi) = (P_{id}.P_{id}) + P_{iE}.([R(Z_i, \phi)]^T[[R(Z_i, \phi)]P_{iE}) - 2P_{id}([R(Z_i, \phi)]P_{iE}) \qquad (2.7)$
$\Rightarrow \Delta P(\phi) = (P_{id}.P_{id}) + (P_{iE}.P_{iE}) - 2P_{id}([R(Z_i, \phi)]P_{iE}) \qquad (2.8)$
Avec $(P_{id}.P_{id})$ et $(P_{iE}.P_{iE})$ sont des constantes positives.
Donc minimiser $\Delta P(\phi)$, revient a minimiser

$$G_1(\phi) = P_{id}([R(Z_i, \phi)]P_{iE} \qquad (2.9)$$

De la même façon, les vecteurs d'orientations deviennent :
$E'_j = [R(Z_i, \phi)]E_j$ pour $j = 1, .., 3$
et l'erreur d'orientation s'écrit alors :

$$\Delta O(q) = \sum_{j=1}^{3} \sigma_j (d_j . E'_j(\phi) - 1)^2 \qquad (2.10)$$

avec $d_j . E'_j(\phi) = cos\psi_j(\phi)$
de là l'erreur d'orientation deviens :

$$\Delta O(q) = \sum_{j=1}^{3} \sigma_j (cos\psi_j(\phi) - 1)^2 \qquad (2.11)$$

La figure suivante résume ce qui a précédé [27] :

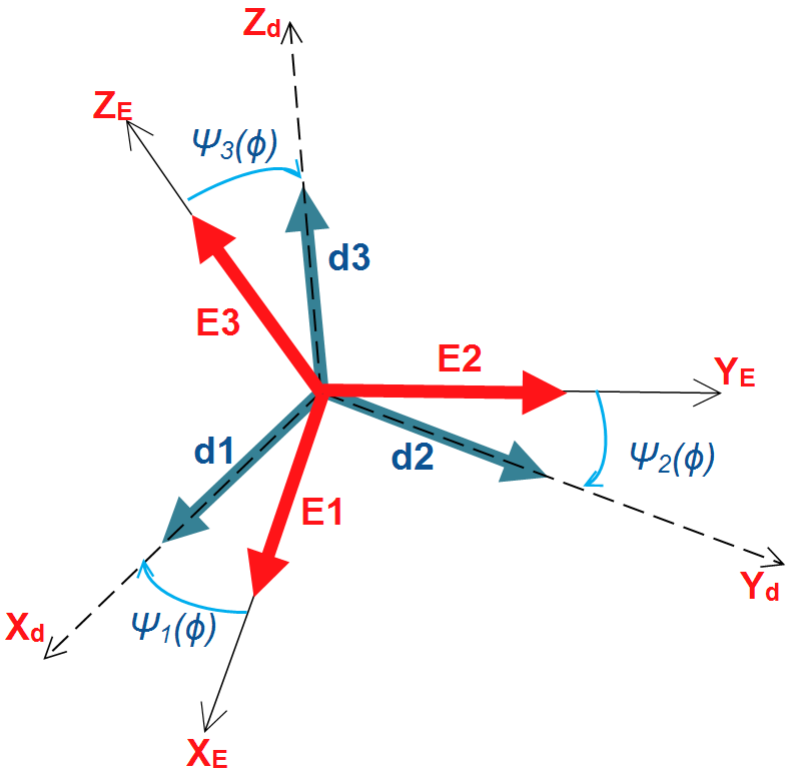


FIGURE 2.2.5: Définition des angles de direction

Sachant que $\Delta O(q)$ est minimisé quand $cos\psi_j(\phi) = 1$, c'est-à-dire quand l'angle $\psi_j(\phi)$ tend vers zéro pour tous les axes.

Et on sais que le cosinus est borné en -1 et $+1$, alors minimiser $\Delta O(q)$, revient a maximiser

$$G_2(\phi) = \sum_{j=1}^{3} cos\psi_j(\phi) \qquad (2.12)$$

En combinant les deux fonctions G_1 et G_2 , on aura les fonctions objectives suivantes :

$$G(\phi) = W_p G_1(\phi) + W_O G_2(\phi) \qquad (2.13)$$

où $W_p$et W_O sont des réels positifs qui représentent respectivement les poids de position et d'orientation.

2.2.2.3 Problème 3 : Cinématique-Maximisation

Le problème 3 est de trouver$\phi^\star$ qui maximise la fonction $G(\phi^\star)$
avec $\underline{\phi} \leqslant \phi \leqslant \overline{\phi}$
$\underline{\phi}$: borne inférieure de l'angle θ_i.
$\overline{\phi}$: borne supérieure de l'angle θ_i.
Selon la formule de Rodrigues. on peut montrer que :

$$[R(Z_i, \phi)P_{iE}] = Z_i(P_{iE}.Z_i)(1 - cos\phi) + P_{iE}cos\phi + (Z_i \wedge P_{iE})sin\phi \qquad (2.14)$$

avec $\wedge$le produit vectoriel.
La fonction $G_1(\phi)$ devient :
$G_1(\phi) = P_{id}[Z_i(P_{iE}.Z_i)(1 - cos\phi) + P_{iE}cos\phi + (Z_i \wedge P_{iE})sin\phi] \qquad (2.15)$
$\Rightarrow G_1(\phi) = (P_{id}.Z_i)(P_{iE}.Z_i)(1 - cos\phi) + (P_{id}.P_{iE})cos\phi + P_{id}(Z_i \wedge P_{iE})sin\phi \qquad (2.16)$
et de même pour $G_2(\phi)$:
$G_2(\phi) = \sum_{j=1}^{3} \sigma_j[(d_j.Z_j)(E_j.Z_j)(1 - cos\phi) + (d_j.E_j)cos\phi + dj(Z_J \wedge E_j)sin\phi] \qquad (2.17)$
et la nouvelle fonction objective est obtenue en additionnant $G_1(\phi)$ et $G_2(\phi)$ de façon a avoir :

$$G(\phi) = K_1(1 - cos\phi) + K_2 cos\phi + K_3 sin\phi \qquad (2.18)$$

où
$K_1 = W_p(P_{id}.Z_i)(P_{iE}.Z_i) + W_O \sum_{j=1}^{3} \sigma_j[(d_j.Z_j)(E_j.Z_j) \qquad (2.19)$
$K_2 = W_p(P_{id}.P_{iE}) + W_O \sum_{j=1}^{3} \sigma_j[(d_j.E_j) \qquad (2.20)$
$K_3 = Z_i[W_p(P_{iE} \wedge P_{id}) + W_O(E_j.d_j)] \qquad (2.21)$
et la solution maximale est alors obtenue en posant

$$\frac{dG(\phi)}{d\phi} = (K_1 - K_2)sin\phi + K_3 cos\phi = 0 \qquad (2.22)$$

et

$$\frac{d^2G(\phi)}{d\phi^2} = (K_1 - K_2)cos\phi - K_3 sin\phi < 0 \qquad (2.23)$$

d'où

$$\phi^\star = \tan^{-1}(K_3/(K_2 - K_1)) \qquad (2.24)$$

avec la nouvelle variable d'articulation
$q_i{}^{nouv} = qi + \phi$
Les limites sur les articulations deviennent
$\underline{q_i} \leq q_i + \phi \leq \overline{q_i}$ i=1,...,n
Et pour respecter ces limites :
– $\phi^\star = \overline{q_i} - q_i$ si $\phi^\star > \overline{q_i} - q_i$
– $\phi^\star = \underline{q_i} - q_i$ si $\phi^\star < \underline{q_i} - q_i$

Cas d'une articulation prismatique

Ici, on ne va réduire que l'erreur de position car l'orientation de l'objet final ne dépend pas de la variable S_i
Notons :
S_{iC} : La valeur actuelle de S_i.
λ : Changement de la variable S_i
Alors l'erreur de position s'écrit
$\Delta P(\lambda) = (P_{id} - (P_{iE} + Z_i\lambda)).(P_{id} - (P_{iE} + Z_i\lambda)) \qquad (2.25)$
$\Rightarrow \Delta P(\lambda) = \varsigma P - \varsigma P - 2(\varsigma P.Z_i)\lambda + \lambda^2 \qquad (2.26)$
avec $\varsigma P = P_{id} - P_{iE}$
Et le problème sera de trouver $\lambda^\star$ qui minimise l'erreur de position
Avec

$\underline{\lambda} \leqslant \lambda \leqslant \overline{\lambda}$

$\underline{\lambda} = S_{iC} - \overline{S_i}$ et $\overline{\lambda} = \overline{S_i} - S_{iC}$
$\frac{\Delta P(\lambda)}{\Delta\lambda} = 2\lambda - \varsigma P Z_i$
et
$\frac{d^2\Delta P(\lambda)}{\Delta\lambda^2} = 2 > 0$
S'il n'y a pas de contrainte sur λ alors, il existe une seule $\tilde{\lambda}$ qui minimise $\Delta P(\lambda)$
avec $\tilde{\lambda}$:$\varsigma P.Z_i$
$\underline{\lambda} \leqslant \tilde{\lambda} \leqslant \overline{\lambda}$ alors $\lambda^\star = \tilde{\lambda}$
si :
$\Delta P(\overline{\lambda}) > \Delta P(\underline{\lambda})$,$\lambda^\star = \underline{\lambda}$
sinon
$\Delta P(\underline{\lambda}) > \Delta P(\overline{\lambda})$,$\lambda^\star = \overline{\lambda}$

2.3 FABRIK *(Forward And Backward Reaching Inverse Kinematics)*

Cette section est largement inspirée des travaux de Andreas Aristidou et Joan Lasenby apparus dans [16].

FABRIK est une nouvelle méthode qui permet de résoudre différent problèmes de la cinématique inverse pour plusieurs cas de figure, que ça soit pour des systèmes simples ou bien au contraire pour des systèmes complexes avec une ou plusieurs cibles à atteindre. Cette méthode nous permet d'avoir des postures après calculs assez convaincantes, du fait où on obtient des mouvements lisses sans discontinuités et avec beaucoup de réalisme. Les algorithmes présentés ci-après utilisant cette méthode, vont résoudre les différents cas figure incluant plusieurs fin d'effecteurs avec plusieurs cibles et en présence de forte restrictions au niveau des joints.

2.3.1 Mode de fonctionnement

FABRIK comme son nom l'indique commence du dernier joint vers le premier les ajustant un après un, puis il fait l'opération inverse de la base allant vers l'effecteur de fin, cette méthode ; à l'inverse des autres, se base sur le principe de trouver un point sur une ligne et non de calculer des angles.

2.3.1.1 Algorithme

L'algorithme suivant [16] nous donne en détails le mode de fonctionnement d'une itération complète dans le cas le plus simple c'est-à-dire une chaine articulée avec une cible et sans la présence de restrictions.

Algorithme 2.1 Itération complète de FABRIK

Entrées :
Les positions des joints P_i pour $i = 1, ..., n$.
La position de la cible t
Les distances entre les joints $d_i =\mid P_{i+1} - P_i \mid$ for $i = 1, ..., n-1$
Sorties :
Les nouvelles positions des joints P_i for $i = 1, ..., n$.

1.1 % La distance entre la base et la cible
1.2 $dist =\mid P_1 - t \mid$
1.3 % Vérifie si la cible est atteignable.
1.4 si $dist > d_1 + d_2 + ... + d_{n-1}$ alors % La cibles n'est pas atteignable
1.6 pour $i = 1, ..., n-1$ faire
1.7 % Trouver la distance r_i entre la cible t et les positions P_i
1.8 $r_i =\mid t - P_i \mid$
1.9 $\lambda_i = d_i / r_i$
1.11 $P_{i+1} = (1 - \lambda_i) \cdot P_i + \lambda_i t$ % Trouver les nouvelles positions P_i.
1.12 fin
1.13 sinon
1.14 % La cible est atteignable ; alors, définir b La position initiale de P_1
1.15 $b = P_1$
1.16 % Vérifier si la distance entre l'effecteur de fin P_n et la cible t est plus grande à une certaine tolérance.
1.17 $difA =\mid P_n - t \mid$
1.18 tant que $difA > tol$ faire
1.19 % ETAPE 1 : Déroulement vers l'avant (FORWARD REACHING)
1.21 $P_n = t$ % fixer l'effecteur de fin comme étant la cible
1.22 Pour $i = n-1, ..., 1$ faire
1.23 % Trouver la distance r_i entre les nouvelles positions P_{i+1} et les positionsP_i
1.24 $r_i =\mid P_{i+1} - P_i \mid$
1.25 $\lambda_i = d_i / r_i$
1.27 $P_i = (1 - \lambda_i) \cdot P_{i+1} + \lambda_i P_i$ % Trouver les nouvelles positions P_i.
1.28 fin
1.29 % ETAPE 2 : Déroulement vers l'arrière (BACKWARD REACHING)
1.31 $P_1 = b$ % fixer P_1 comme postition initial.
1.32 Pour $i = 1, ..., n-1$ faire
1.33 % Trouver la distance r_i entre les nouvelles positions P_i et les positions P_{i+1}
1.34 $r_i =\mid P_{i+1} - P_i \mid$
1.35 $\lambda i = d_i / r_i$
1.36 %Trouver les nouvelles positions P_i.
1.37 $P_{i+1} = (1 - \lambda_i) \cdot P_i + \lambda_i P_{i+1}$
1.38 fin
1.39 $difA =\mid P_n - t \mid$
1.40 fin
1.41 fin

2.3.1.2 Explications

On a comme entrées :
- P_i : les joints de positions de la chaine articulée avec $i = 1, ..., n$. P_1 : le joint de base et Pn : l'effecteur de fin.
- t : la cible qu'on veut atteindre.
- b : la position de base.

Les sorties sont représentées comme les nouvelles positions des joints de P'_i avec $i = 1, ..., n$.

Avant tout on va calculer les distances d_i entre chacun des joints avec $d_i =\mid P_{i+1} - P_i \mid$, pour $i = 1, \ldots, n-1$.

Afin de savoir si la cible t est atteignable ou pas, on calcule la distance entre le joint de base et la cible t, $dist =\mid P_1 - t \mid$, et si la distance est inferieure à la somme totale des distances d_i, $dist < \sum_{1}^{n-1}(d_i)$, alors la cible est atteignable, dans le cas contraire, $dist > \sum_{1}^{n-1}(d_i)$ alors la cible n'est pas atteignable et l'algorithme s'arrête.

Si la cible est atteignable alors une itération complète est lancée, cette itération se déroule en deux étapes. La première étape consiste à déterminer les positions de tous les joints en commençant par l'effecteur de fin P_n, jusqu'au joint de base P_1, en premier lieu l'effecteur de fin va devenir la cible t, c'est-à-dire qu'on va projeter temporairement P_n sur la position de la cible t, $P'_n = t$. après il va tracer une ligne l_{n-1}, qui va passer par P'_n et P_{n-1} et finalement P'_{n-1} va être sur cette ligne avec une distance d_{n-1} de P'_n. La même opération est répétée pour les autres joints jusqu'à arriver à la base P_1 qui prend bien sûr une nouvelle position P'_1, sachant que la position de base $b = P_1$ doit rester la même (elle est fixe), elle doit donc revenir à sa position initiale et pour ce, il faut procéder à la deuxième étape de l'algorithme.

Dans cette deuxième étape, on procède de la même manière que l'opération précédente mais en commençant de la base en allant vers l'effecteur de fin, c'est-à-dire qu'on va fixer $P'_1 = b$ (la position de base initiale) donc P'_1 va être projeté vers b et on va obtenir la position $P^{"}_1$ puis une ligne l_1 va être tracé et va passer de $P^{"}_1$ vers P'_2 alors la nouvelle position de P'_2 qui est $P^{"}_2$ va se mettre sur cette ligne l_1 d'une distance d_1 de $P^{"}_1$. Ainsi une itération complète s'est achevée.

Apres avoir obtenu une itération complète, on vérifie si l'effecteur de fin à atteint sa cible, qu'il soit exactement sur elle ou suffisamment proche (selon des critères prédéfinie), et si ce n'est pas le cas alors d'autres itérations ce mettront en marche afin d'atteindre la cible. Généralement, la version de FABRIK pour un système sans contraintes, converge toujours vers notre résultat, bien sur tant que la somme de la longueur des chaines est supérieure à la distance entre le joint de base et la cible.

Dans le cas contraire où la cible n'est pas atteignable il y'a quand même un critère de fin pour l'algorithme, ce critère de fin peut être définie par une comparaison entre la position initiale et actuelle de l'effecteur de fin et cette distance entre les deux positions

doit être inferieur à une certaine tolérance prédéfinie.

Dans le cas extrême où la cible n'a pas pu être atteinte et où le nombre d'itérations deviens conséquent alors on fixe un nombre d'itération à ne pas excéder et l'algorithme est brusquement arrêté.

2.3.1.3 Exemple

La figure (2.3.1) suivante résume le fonctionnement de l'algorithme (2.1) :

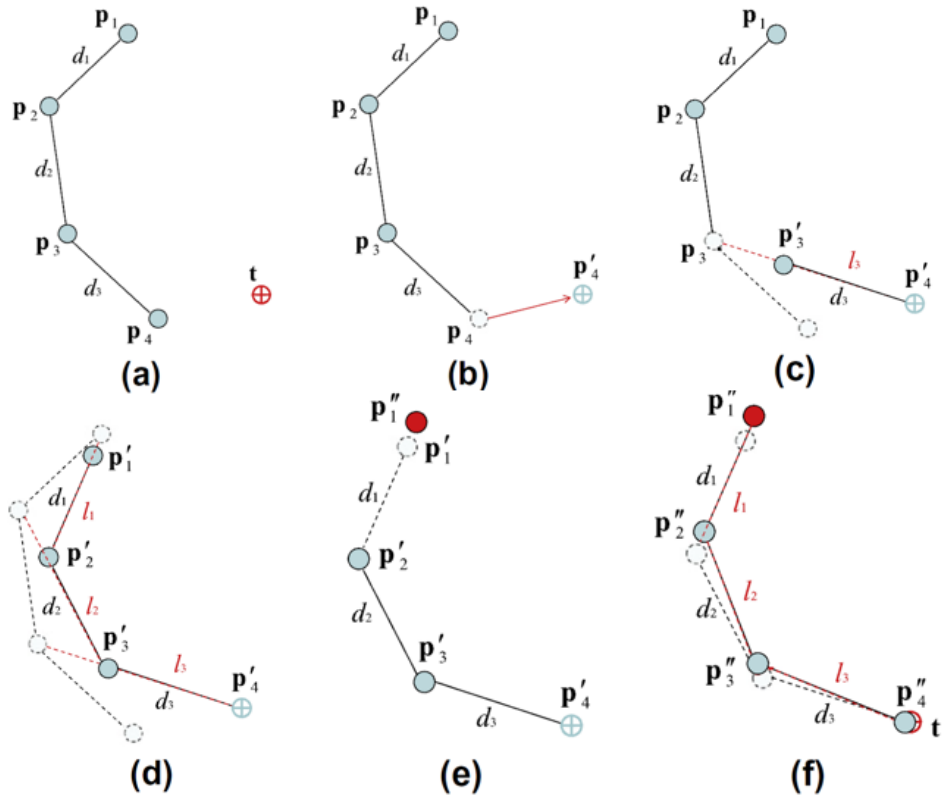


FIGURE 2.3.1: Itération complète de FABRIK

Dans cet exemple on a une chaine articulée qui comporte 4 joints ($P_i = P_1, \ldots, P_n$) et où l'effecteur de fin doit atteindre une cible t .

(a) : Position au repos de la chaine.

(b) : La première étape de l'algorithme est mise en marche : Déplacement de P_4 sur t.

(c) : La nouvelle position de P_3 est sur la ligne l_3 qui passe par P'_4 et P_3 et qui est d'une distance d_3 de P'_4.

(d) : L'opération est répétée pour tous les joints.

(e) : La deuxième étape de l'algorithme commence : déplacement de P'_1 sur $P^{"}_1$ et qui est la position initiale b.

(f) : L'opération est répétée mais cette fois ci du sens inverse.

Il est important qu'une méthode de cinématique inverse puisse résoudre un cas plus complexe et qui reflète la réalité comme des parties du corps humain comme une main ou une épaule, qui peuvent se composées de plusieurs effecteurs de fin et/ou de plusieurs cibles à atteindre et aussi de plusieurs sous-bases qui relient plusieurs chaines entre elles.

La figure (2.3.2) illustre un exemple de présence de plusieurs sous-bases.

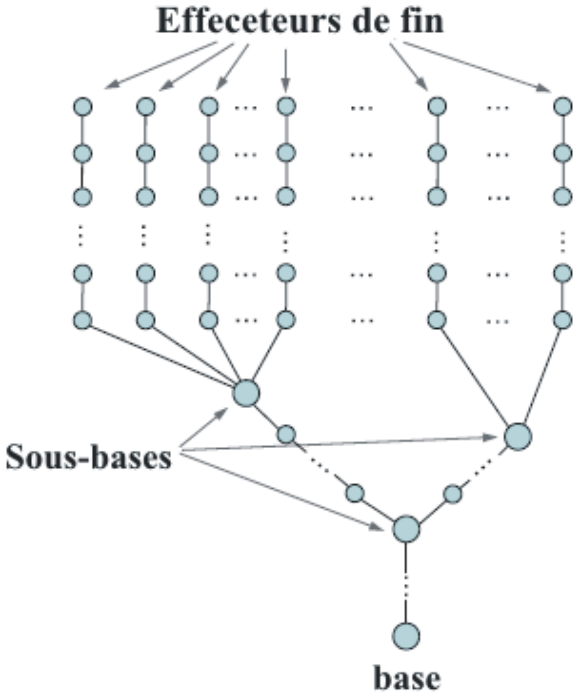


FIGURE 2.3.2: Chaine avec plusieurs sous-bases.

Le principe de FABRIK pour un cas complexe est pratiquement le même que pour un cas simple c'est-à-dire que pour un système composé de plusieurs chaines reliées à une sous base l'algorithme va appliquer la première étape sur chacune des chaines, des effecteurs de fin vers leur base parente et qui est une sous base partageant avec d'autres sous base une autre base parente et il va continuer a traité ainsi autant de sous bases intermédiaires jusqu'à arriver à la base principale, alors la deuxième étape pourra enfin commencer et qui consiste à faire l'opération inverse, traitant toutes les chaines séparément en allant de la base principale vers les sous bases, puis des sous bases vers les effecteurs de fin qui pourront atteindre leur cibles, le nombre d'opérations est répété autant de fois que nécessaire on peut avoir donc plus d'une itération.

2.3.2 FABRIK avec contraintes :

La plupart des parties du corps humain présentent bien évidemment des restrictions biomécaniques naturelles, donc pour la représentation de modèles de certaines parties du corps humain, il est primordial d'introduire des contraintes pour obtenir un certain réalisme. Beaucoup de paramètres entrent en jeu afin de pouvoir représenter au mieux un modèle correcte, ces paramètres vont déterminer l'espace de mouvement et la structure du modèle.

Les joints sont les principaux paramètres qui décrivent la structure d'un modèle, ainsi on peut prendre comme exemple le modèle de l'épaule qui comporte 3 joints ou bien la colonne vertébrale qui est représentée par 24 joints.

2.3.2.1 Contrainte sur le joint

Un joint est définie dans l'espace de mouvement par sa position et son orientation qui sont possible par son nombre de degrés de liberté, pour ce qui suit on va se concentrer sur le cas le plus général où le joint possède 3 degrés de liberté, 2 degrés de rotation simple qui représentent la possibilité à la chaine d'atteindre le vecteur de destination et 1 degré de liberté qu'on appelle d'orientation qui va faire une rotation autour de ce vecteur.

Donc ces 3 degrés de liberté vont représenter les restrictions sur les joints

Prenons comme exemple de joint, une rotule avec une limite d'orientation représentée par le rotor R, et une limite de rotation représentée par θ_1,θ_2,θ_3 et θ_4, Alors on obtient un cône qui va décrire les restrictions de rotation sur le joint comme illustré dans la figure (2.3.3) suivante :

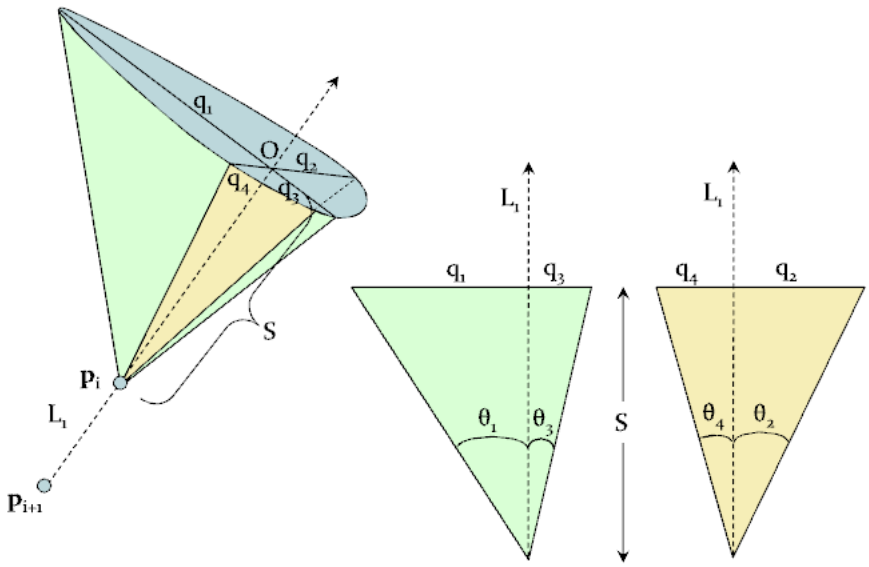


FIGURE 2.3.3: Cone représentant Les restrictions de rotations.

avec

S : Distance entre le joint et le centre de l'ellipse.

q_j : $j = 1, ..., 4$. avec $q_j = S.tan(\theta_i)$

et le champ de mouvement est représenté en bleu.

On peut avoir 3 différentes sortes de représentation conique selon les angles θ_i comme montrer sur la figure ci-dessous :

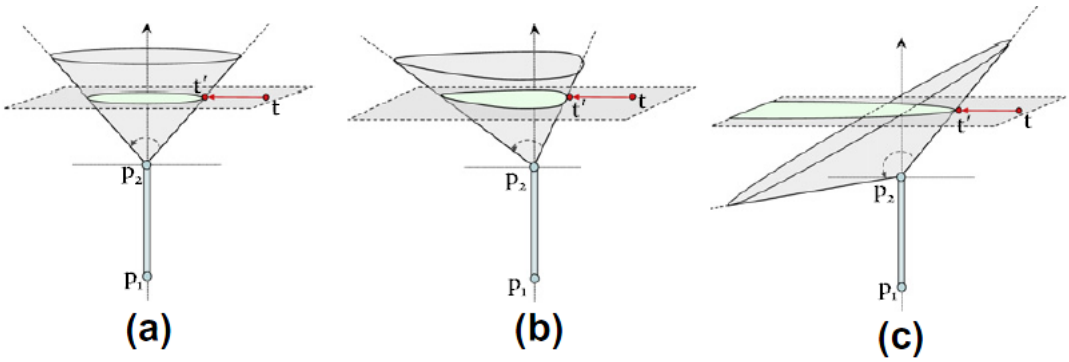


FIGURE 2.3.4: Différentes sortes de cônes

(a) : Les θ_i sont équivalents alors on obtient un cercle.

(b) : Les θ_i sont différents alors on obtient une forme ellipsoïdale.

(c) : Deux θ_i seulement sont équivalents et sont supérieur ou inferieur à 90° alors on obtient une parabole.

2.3.2.2 Algorithme pour les restrictions d'orientation :

Imaginons qu'on soit dans la première étape de l'algorithme (2.1) citée précédemment, précisément à l'instant où on a trouvé la nouvelle position du joint $(i-1)^{ième}$, la démarche de l'algorithme pour une restriction d'orientation est décrite comme suit [16] :

Algorithme 2.2 Restrictions sur l'orientation

Entrées :
Le rotor R qui exprime les restrictions d'orientation sur P_i
Sorties :
La nouvelle orientation de P_i et qui est P'_i
2.1 Vérifier si le rotor est dans la zone possible de mouvement
2.2 Si oui Alors
2.3 ne rien faire et quitter
2.4 Sinon
2.5 réorienter le joint P'_i de façon a ce que le rotor R sois dans le dans le champ de mouvement.
2.6 Fin

On va d'abord détecter la restriction d'orientation appliquée sur le joint, de là on va vérifier l'orientation du joint, si elle est en dehors des limites mouvement possible, on va réorienter le joint de façon à rester dans les limites prédéfinies.

Exemple :

Voici la figure (2.3.5) suivante :

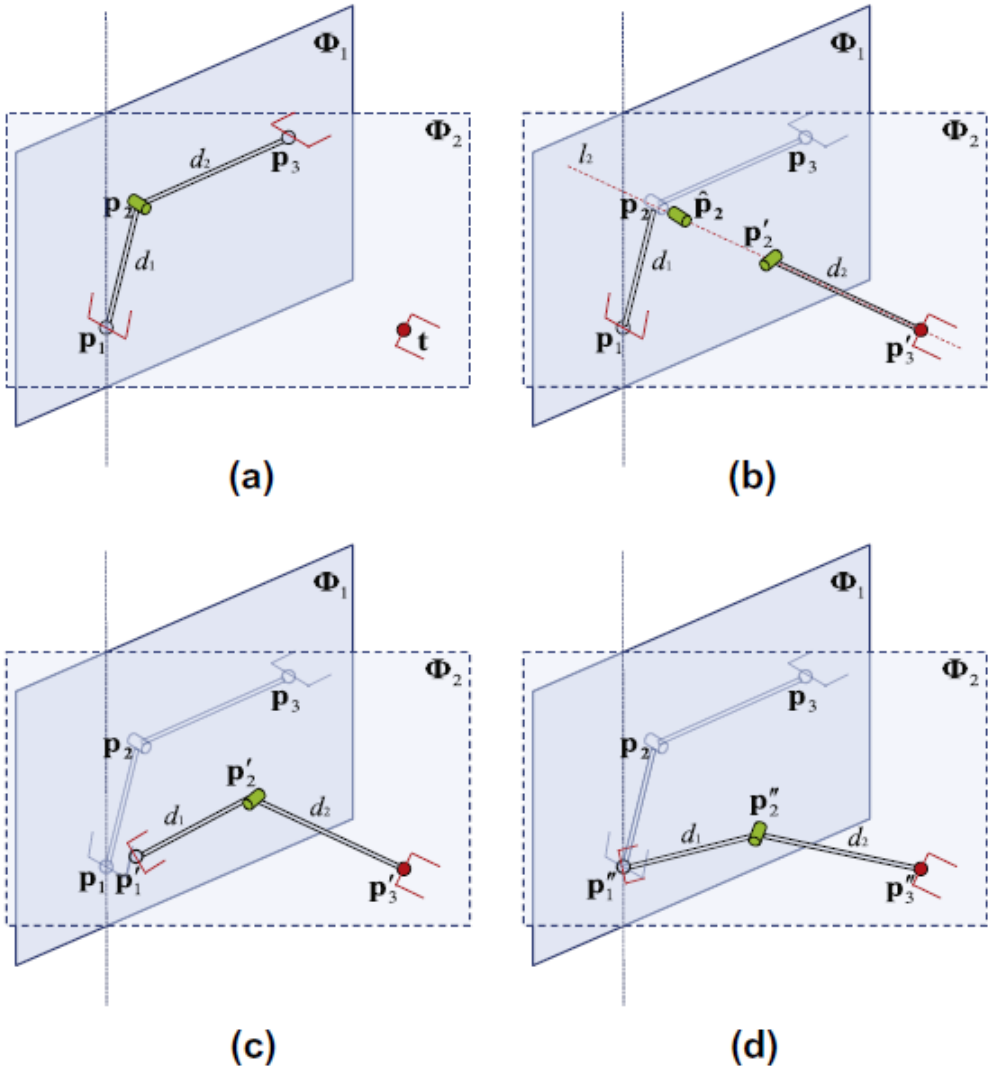


FIGURE 2.3.5: Contrainte d'ortientation

(a) : La position initiale du bras manipulateur et de la cible t. Le plan Φ_1 représente le plan d'orientation pour P_2(un degré de liberté),le plan Φ_2 représente la position de la cible.

(b) : Projection et orientation de P_3 sur la cible t, puis projection de P_2 sur Φ_2 obtenant $\hat{P}_2$ après, tracer la ligne l_2 qui passe par P'_3 et $\hat{P}_2$ et sur laquelle va se positionner P'_2 à une distance d_2 de P'_3. Réorienter P'_2 en appliquant les contraintes d'orientations.

(c) : Trouver P'_1 qui se trouve sur la ligne l_1 qui passe par P_1 et P'_2, avec une distance d_1 de P_1.

(d) : Maintenant que tout est sur le plan Φ_2 donc le problème devient un problème en 2D, alors FABRIK Peut être utilisé dans sa forme la plus simple.

2.3.2.3 Algorithmes pour les restrictions de rotation :

On va expliquer la méthode Fabrik avec des restrictions de rotation en deux parties.

La première partie : Déroulement vers l'avant *(Forward reaching)*.

Algorithme (2.3) [16] :

Algorithme 2.3 Forward reaching avec restrictions de rotation

Entrées :
La position actuelle de la cible t
Les θ_i qui determinent le cône
d_i les distances entre P_i et P_{i-1}.
Sorties :
Les nouvelles positions des joints p_i pour $i = 1, ..., n$.

3.1 Tracer la ligne l_1
3.2 Trouver O qui est la projection de la cible sur la ligne l_1
3.3 O est l'origine des axes y et $x \Rightarrow$ passage au plan simplifier en 2D
3.5 Trouver les nouvelles positions de P_i , notées P'_i
3.6 Vérifier si P'_i est dans le champ de mouvement, Alors
3.7 Ne rien faire
3.8 Sinon
3.9 Projeter P_i sur l'ellipse et voir dans quel quadrant est le point plus proche, $\hat{P}_i$
3.10 Translater $\hat{P}_i$ sur la ligne du cône d'une distance d_i.
3.11 P'_i la nouvelle position de P_i
3.12 fin

On a comme données :

La position de la cible t que la quelle on va projeter le joint P_i.

La distance d_{i-2} qui est la distance entre le joint P_{i-1} et P_{i-2}.

On suppose des restrictions de rotation entre les joints P_i et P_{i-1}, qui sont représentés par un cône limité pas les angles θ_i.

Dans un premier temps On va projeter la cible t sur la ligne l_1 qui passe par P_i et P_{i-1}, et sur cette ligne ; d'une distance S de P_{i-1}, on obtient l'axe déterminant le cône.

Enfaite ce cône qui a comme origine le joint P_{i-1}, est la restriction de mouvement sur le joint P_{i-2}.

On va vérifier que P_{i-2} est dans le champ de mouvement possible si ce n'est pas le cas et qu'il est en dehors du cône, alors on va fixer P_{i-2} sur le point le plus proche du cône et on va le translater afin qu'il soit d'une distance d_{i-2} du joint P_{i-1}. On va suivre la même procédure pour la suite.

Localisation du point projeté sur l'ellipse

Voici la figure (2.3.5) suivante :

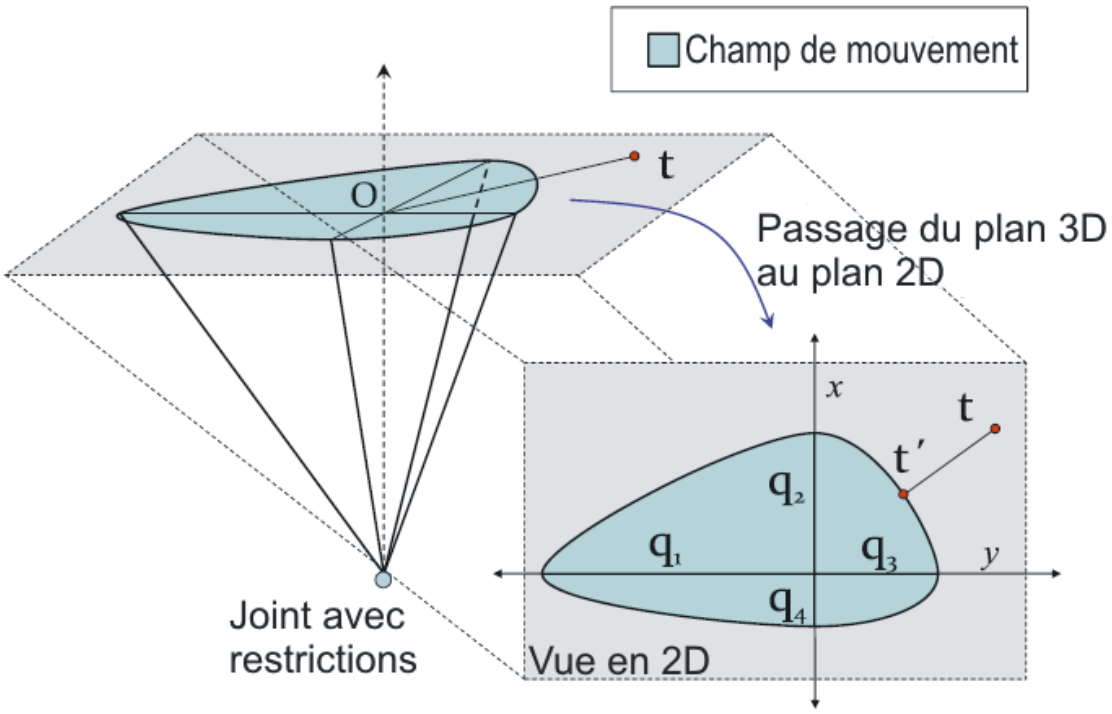


FIGURE 2.3.6: Localisation du point sur l'ellipse

Comme monter dans la figure (2.3.5) pour pouvoir étudier le nouvel emplacement du joint sur le cône il faut :

- Simplifier la vue en passant d'une vue 3D a une vue 2D.
- Localiser sur quel quadrant se trouve la projection du joint.

Deuxième partie : déroulement vers l'arrière *(Backward reaching)*.

Ici, on fait l'opération inverse *(voire algorithme 2.1)* joint par joint jusqu'à arriver à la cible en vérifiant toujours les contraintes de rotation qu'on va anticiper de la même manière que dans l'algorithme (2.3).

Si il y'a des restrictions de rotation sur le dernier joint et qui ne peut pas atteindre la cible alors on projette la cible sur le point le plus proche du cône comme montrer dans la figure ci-dessous :

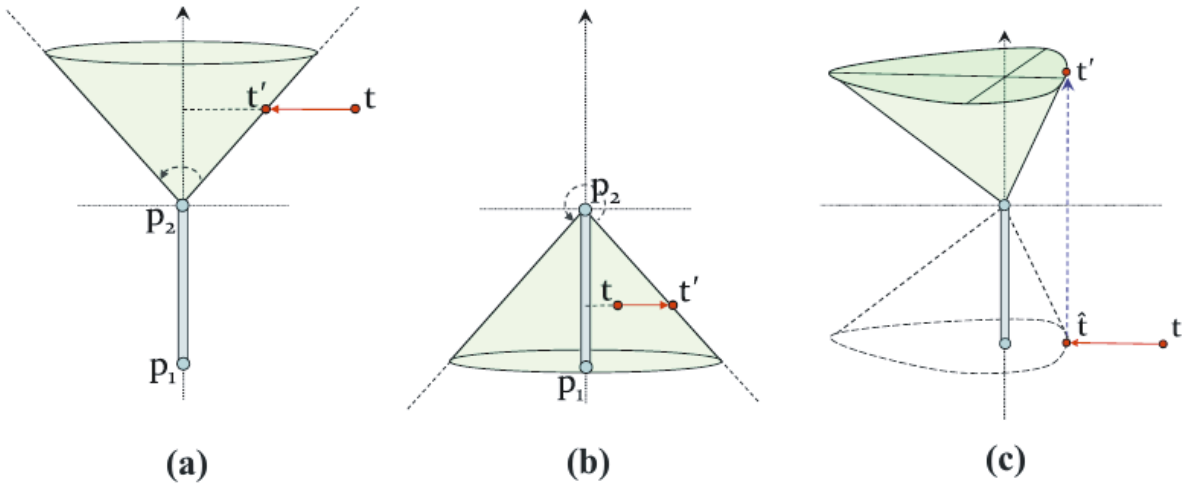


FIGURE 2.3.7: Les possibilités de positionnement pour t

D'après la figure, On distingue trois cas possible :
(a) La cible est projetée sur le point le plus proche du cône.
(b) Le plan de restriction dépasse les 180 degrés.
(c) La cible est localisée dans un hémisphère diffèrent du cône.
Et pour finir on refait la même opération autant de fois que nécessaire jusqu'à atteindre la cible (convergence).
L'algorithme (2.4) suivant résume ce qui a été dit plutôt :

Algorithme 2.4 Backward reaching avec restrictions de rotation

Entrées :
La position de base b
d_i les distances entre P_i et P_{i-1}.
Sorties :
La nouvelle position temporaire de t
4.1 Vérifier si il y'a des restrictions de rotation sur P_i
4.2 Si oui
4.3 Voir si la cible t est atteignable
4.5 si oui, ne rien faire
4.6 sinon
4.7 Projeter la cible sur le point le plus proche du cône
4.8 fin

2.3.2.4 Exemple

Voici la figure (2.3.8) qui présente un exemple d'application de FABRIK sur un bras qui comporte des contraintes d'orientation et de rotation.

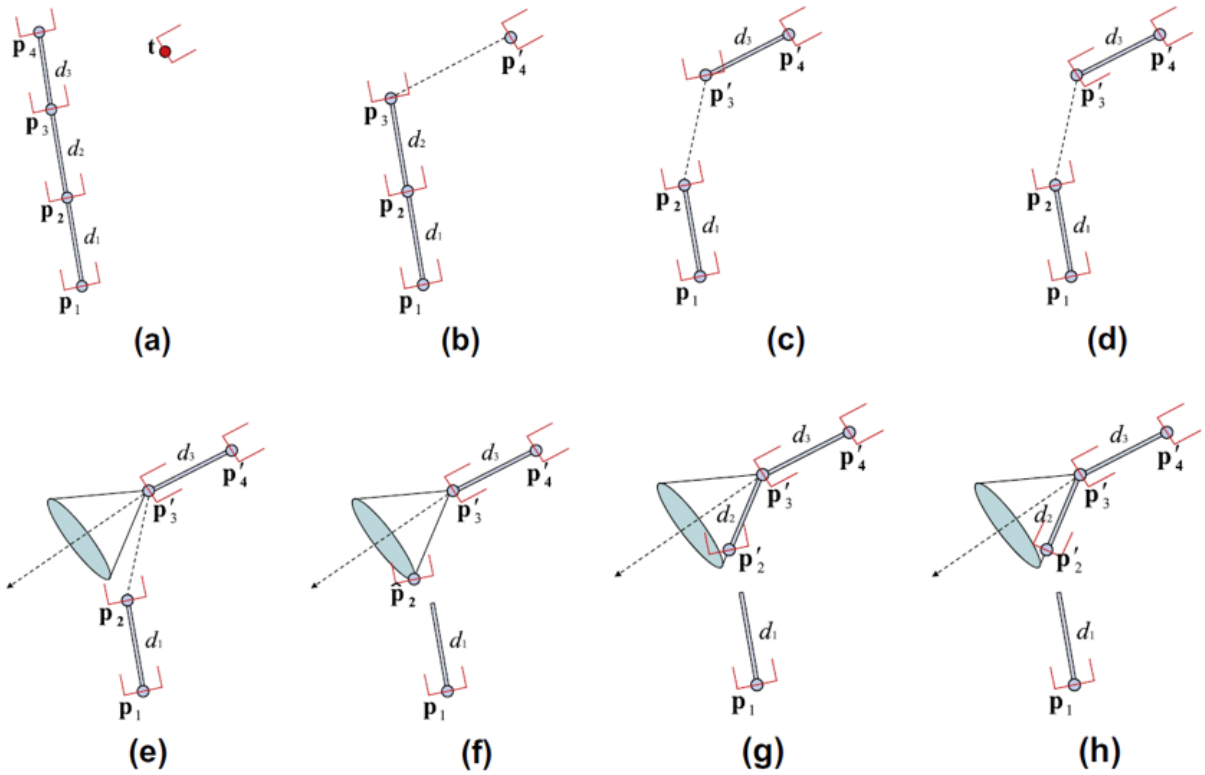


FIGURE 2.3.8: FABRIK avec des contraintes d'orientation et de rotation

(a) : Configuration initiale du bras et de la cible t.
(b) : Relocaliser et réorienter P_4 sur la cible t.
(c) : Déplacer le joint P_3 à P'_3 sur la ligne qui passe par les points P'_4 et P_3 et d'une distance d_3 de P'_4 .
(d) : Réorienter le joint P'_3 de façon à ce que le rotor exprimant possibilités d'orientation entre P'_3 et P'_4, soit conforme aux limites prédéfinies.
(e) : Les limites de rotations sont représentées par un cône qui a comme axe, la ligne qui passe par les points P'_4 et P_3.
(f) : Relocaliser P_2 en $\hat{P}_2$, $\hat{P}_2$ est le point le plus proche appartenant au cône, ainsi $\hat{P}_2$ est dans les limites du champ de mouvement définient par les contraintes. de rotations.
(g) : Déplacer $\hat{P}_2$ à P'_2 pour respecter la longueur du lien.
(h) : Réorienter P'_2 afin de satisfaire les contraintes d'orientation.

2.4 Conclusion

Dans ce chapitre, nous avons exposé les deux méthodes FABRIK et CCD et on a pu constater que ce sont des méthodes qui présentent beaucoup de qualité du point de vue rapidité et précision.

Dans le chapitre suivant, nous allons présenter l'application "HandiAccess" et on va y implémenter la méthode FABRIK et comparer les résultats avec la méthode CCD déjà implémentée.

Chapitre 3

HandiAccess

3.1 Introduction

Ce chapitre contient deux grandes parties :

Dans la première nous allons étudier l'application *HandiAccess* et voir les outils qui ont été utilisés dans sa conception.

Nous allons commencer par décrire les outils de développement, puis nous allons voire les principales bibliothèques qui ont facilité la conception du logiciel qui sont les bibilothèques : *ROBOOP*, *NewMat* ainsi que *Ogre3D* qui est le moteur 3D.

Ensuite nous allons faire une brève présentation de *3d studio max* qui a était utilisée pour la modélisation 3D de l'Avatar ainsi que l'environnement.

Pour terminer on va faire une description de l'application et de ses fonctionnalités.

Dans la deuxième partie nous allons implémenter la méthode FABRIK d'abord sur une chaine articulée et par la suite l'intégrer à l'avatar de HandiAccess puis la comparer avec la méthode CCD et on finira ce chapitre avec l'étude des résultats obtenus.

3.2 Les outils de développement

3.2.1 Le langage de developpement

L'application HandiAccess est développée en C++, le choix du langage s'est porté sur le C++ du fait qu'il est l'un des langages les plus populaires et les plus utilisés, l'avantage qu'il soit très utilisés et qu'il y est une grande communauté active autour qui propose des tutoriels ou de la documentation.

Le C++ est un langage très rapide souvent utilisé pour les applications qui ont besoin de performances ou bien pour des applications fonctionnant en temps réel.

Il existe de nombreuses bibliothèques en C++, qui facilitent la programmation comme des bibliothèques pour la Robotique ou pour le rendu 3D.

3.2.2 Les bibliothèques utilisées

3.2.2.1 La bibliothèque Newmat

Cette bibliothèque est utilisée pour la manipulation des matrices, on retrouve les opérations matricielles standard ainsi que des notions plus avancées comme les moindres carrés ou bien la résolution des valeurs propres[29].

3.2.2.2 La bibliothèque ROBOOP

La bibliothèque ROBOOP est une bibliothèque sous C++ qui est utilisée pour la robotique et appropriée pour la synthèse et la simulation des modèles de robots manipulateurs, elle est composée de plusieurs classes et fait aussi appel à d'autres bibliothèques comme la bibliothèque Newmat [30].

La classe Robot_basic[12]

Cette classe permet la déclaration du modèle du robot, soit en le choisissant à partir des modèles déjà présent dans la bibliothèque ou de le charger à partir d'un fichier de configuration (d'extension ".conf") dans *"link"*.

- La classe kine : Cette méthode surchargée retourne le résultat de la géométrique directe. Elle existe en trois versions : :
  - La fonction *kine(int j)* retourne la position de l'articulation j à partir de la première articulation.
  - La fonction *kine(Matrix &rot, ColumnVector &pos, int j)* retourne la position de l'articulation j à partir de la première articulation dans *"pos"* et la matrice de rotation dans *"rot"*
  - La fonction *kine(void)* retourne la position de l'objet final à partir de la première articulation.
- La fonction inv_kin cette fonction détermine pour un point désiré les variables articulaires correspondantes.

La classe links

Cette classe englobe tout le nécessaire requis pour caractériser un lien comme il est défini dans Denavit et Hartenberg dans la notation standard ou modifiée.

Le tableau (3.1) suivant résume ces données[12] :

	Paramètre	Type	Description du paramètre
géométrique	*joint_type*	int	il est égal à zéro si l'articulation est rotationelle et à 1 si elle est prismatique
	theta, d, a, alpha	Real	Les paramètres DH
	joint_offset	Real	Décalage de l'articulation
	P	ColumnVector	Position de l'articulation
	R	Matrix	Matrice de rotation de l'articulation
	DH	Bool	true = notation standard, false = notation modifiée
	theta_min	Real	Bande inférieure de la variable
	theta_max	Real	Bande supérieure de la variable
inertie	*m*	Real	la masse
	r	ColumnVector	Position du centre de gravité
	I	Matrix	Matrice du tenseur d'inertie
Moteur	*Im*	Real	Inertie de la partie rotor du moteurs
	Gr	Real	Rapport de vitesse
	B	Real	Coefficient du frottement visqueux
	Cf	Real	Frottements de Coulomb

TABLE 3.1: Les paramètres de la classe ″link″

Les bibliothèques dynamiques DLL

Les DLL *(Dynamic Link Library)*, contiennent généralement des fonctions qui peuvent êtres utilisées par plusieurs programmes.

Ce sont des bibliothèques où les fonctions sont appelées pendant l'exécution du programme. L'avantage est donc que l'exécutable est plus léger mais il faut fournir la bibliothèque avec le programme. L'autre avantage des DLL est que si plusieurs programmes ont besoin de la même DLL, ils peuvent tous y accéder alors que la bibliothèque n'est chargée qu'une fois, dans ce cas on parle de bibliothèque partagée et cela permet d'alléger l'utilisation de la mémoire.

3.2.2.3 La bibliothèque Ogre3D

Ogre3D (Object-Oriented Graphics Rendering Engine) est un moteur 3D qui fournit un ensemble de fonctions qui permettent à l'utilisateur, de représenter des objets dans un monde en trois dimensions. Le moteur 3D est un intermédiaire entre l'utilisateur et

les librairies graphiques de plus bas niveau comme DirectX ou OpenGL, qui permet donc de se passer des aspects les plus complexes de la gestion d'objets 3D.

Entre autre Ogre3D va s'occuper de la gestion de la scène donc de l'affichage des personnages, bâtiments et les payasages, la gestion des lumières et des ombres aussi la gestion d'une ou plusieurs caméras [31].

Il utilise aussi TinyXML qui lui permet de charger les modèles 3D et leurs paramètres du document XML.

3.2.3 3D studio max

3D Studio Max est un logiciel de modélisation et d'animation 3D, développé par la société Autodesk et très utilisé dans le monde de l'infographie, il a été utilisé dans la modélisation de l'avatar (squelette et revêtement) ainsi que l'environnement.

La figure (3.2.1) nous montre la modélisation de l'avatar [12].

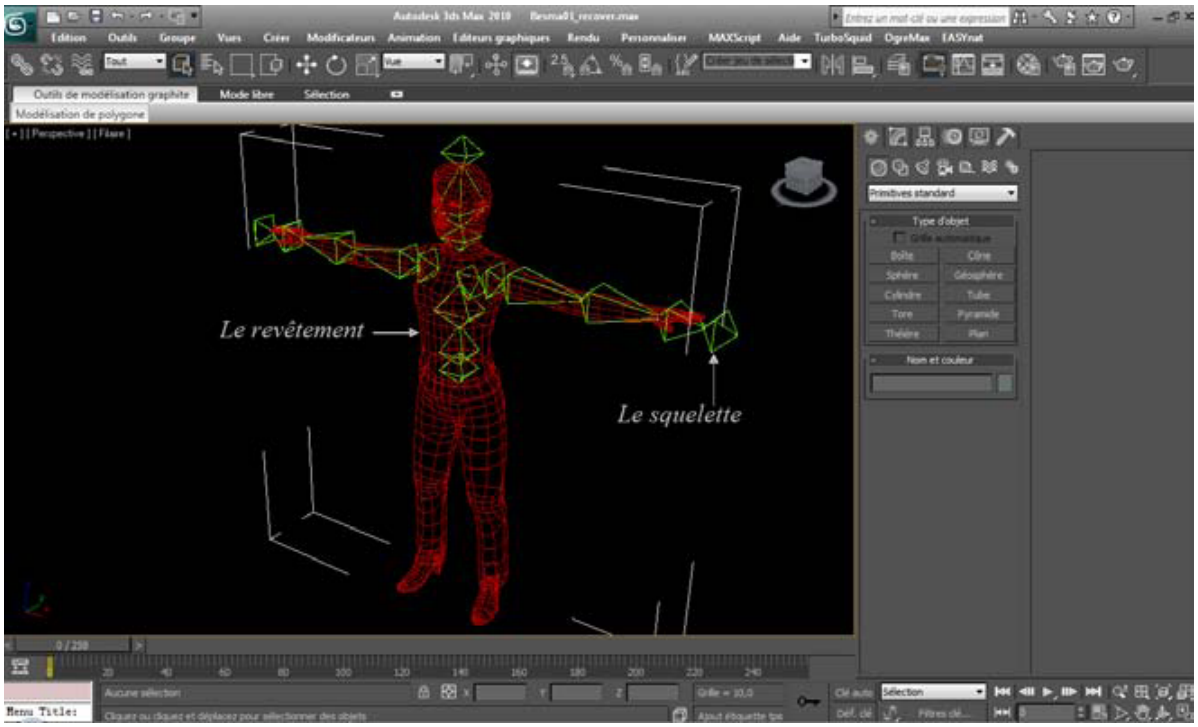


FIGURE 3.2.1: Modélisation 3D de l'avatar

3.3 Description et fonctionnalités

L'application HandiAccess a une interface utilisateur qui comporte une barre d'outils et quatre fenêtres principales.

3.3.1 La barre d'outils

La barre d'outils d'HandiAccess comme pour chaque application englobe toute les fonctionnalités ou options que comporte le logiciel, et elle contient les menus suivants :

Ficher/Edition/Affichage/Aide

Des menus où l'on peut retrouver les options basic d'une application, c'est-à-dire : ouvrir, enregistrer, imprimer, copier...

Modes

Il admet trois modes de fonctionnement de simulation 3D qui sont :

1. Sélection : permet de sélectionner un ou plusieurs objets dans la scène, ce mode est utile par exemple pour appliquer des transformations géométriques sur les objets.
2. Calcul : ici, l'utilisateur peut appeler des algorithmes de planification ou de cinématique inverse.
3. Navigation : Ce mode nous donne la possibilité de naviguer dans le monde 3D, ainsi on peut avancer, reculer, changer le champ et la direction de la vue et aussi changer le mode de visualisation de l'environnement (point, filaire,...).

Transformations

Ce menu compte trois options qui permettent de faire des transformations géométriques sur tout les objets de la scène et qui sont : rotation, translation et scale.

Manual Manipulation

Cette liste déroulante contient deux options :

1. Initialise : Permet d'initialiser l'avatar dans sa position par défaut.
2. Set Target : Positionner la cible manuellement grâce à une boite de dialogue pour y introduire ses coordonnées.

IK

Ce menu propose une liste de quatre options en rapport avec la résolution du problème de la cinématique inverse et qui sont :

1. Solve : Résolution du problème de cinématique proposé
2. Set Avatar Position : Positionner l'emplacement de l'avatar dans l'espace.
3. Set X1 Position : Point par le quel doit passer le mouvement pour éviter un obstacle (cas d'un chemin avec contraintes).
4. Select IK Solver : Faire le choix de la méthode de résolution qui sera utilisé.

Path Planning

Ce menu propose deux options qui permettent de planifier un chemin avec ou sans obstacles (chemin avec ou sans contraintes).

La figure (3.1.1) suivante résume tout ce qui a été dit auparavant :

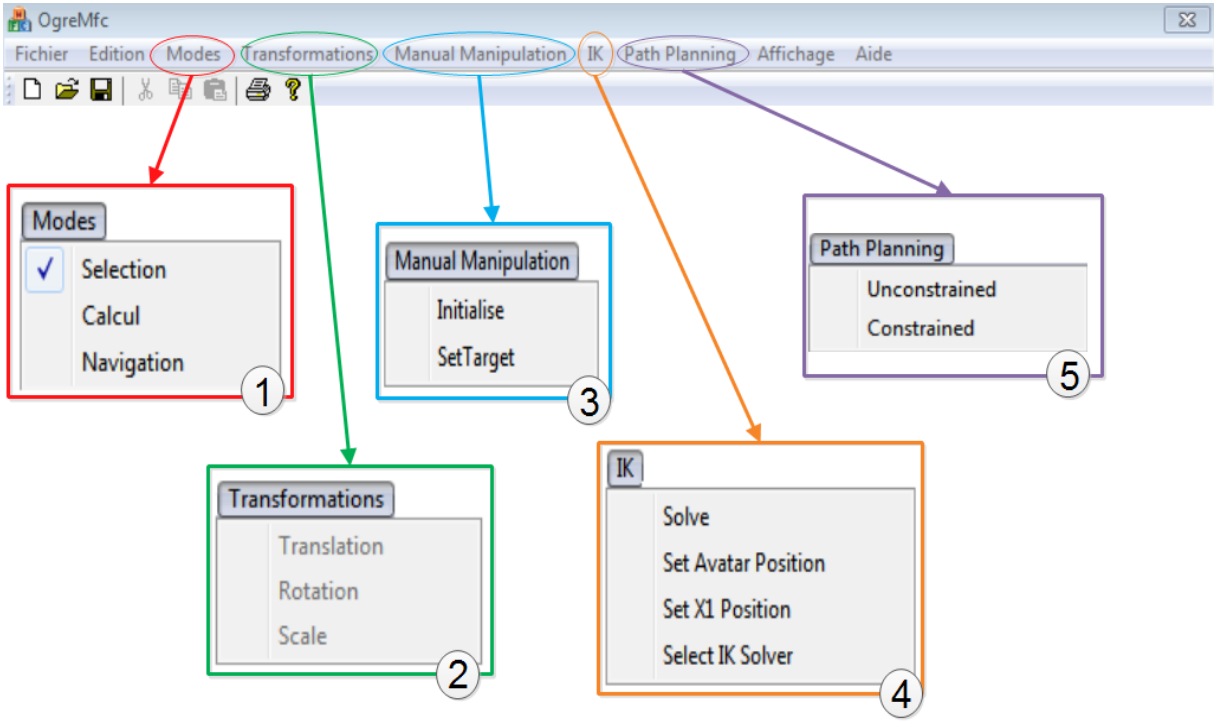


FIGURE 3.3.1: Les menus de la barre d'outils

(1) : Les trois modes de fonctionnement du simulateur.
(2) : Menu déroulant des transformations de base.
(3) : Commande d'initialisation de la posture de l'avatar et de la position de la cible.
(4) : Menu déroulant de cinématique inverse.
(5) : Sélection de la méthode de planification.

3.3.2 L'interface utilisateur

L'interface d'utilisateur se compose de quatre fenêtres :

1. Une fenêtre de visualisation de la scène 3D qui permet de voir toutes les interactions, les modifications ainsi que les résultats obtenus.
2. *Scenes explorer* : Un explorateur qui liste les objets contenus dans la scène 3D, cela concerne les objets 3D, les objets 2D, les sources de lumière et l'avatar.

3. *Display and manual manipulation* : Une boîte de dialogue qui permet une commande directe des 21 degrés de liberté de l'avatar.
4. Une boite de dialogue de transformations qui offre la possibilité de faire des transformations géométriques de base sur les objets : translation, rotation et échelle.

La figure (3.1.2) suivante nous montre les parties décrites ci-dessus :

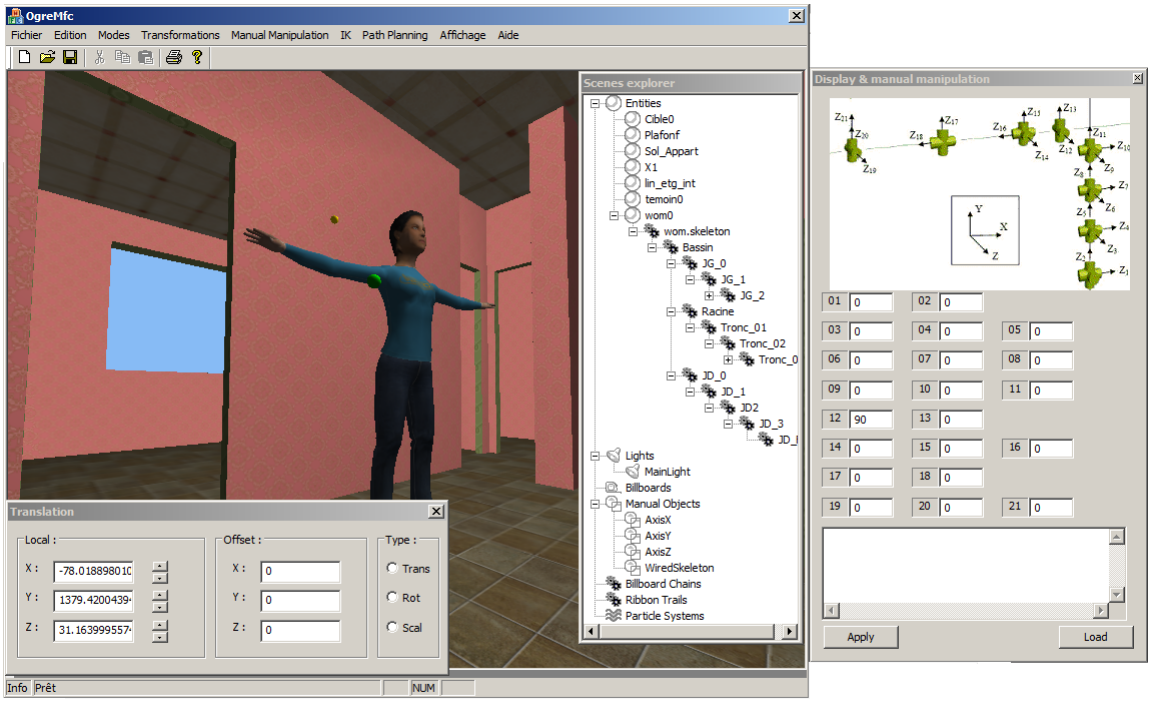


FIGURE 3.3.2: Interface utilisateur

3.4 Résultats

Dans cette section, nous verrons quelques résultats que nous avons obtenus, d'abord en implémentant la méthode FABRIK sans contraintes sur une chaine articulée puis sur notre avatar et le tout sur la plateforme HandiAccess.

3.4.1 Implémentation sur chaine articulée

Nous avons appliqué l'algorithme de FABRIK sur une chaine articulée de neuf degrés de liberté à qui nous avons donné un mouvement en forme de huit. Nous avons obtenu les résultats suivants et sur lesquels on observe des mouvements fluides.

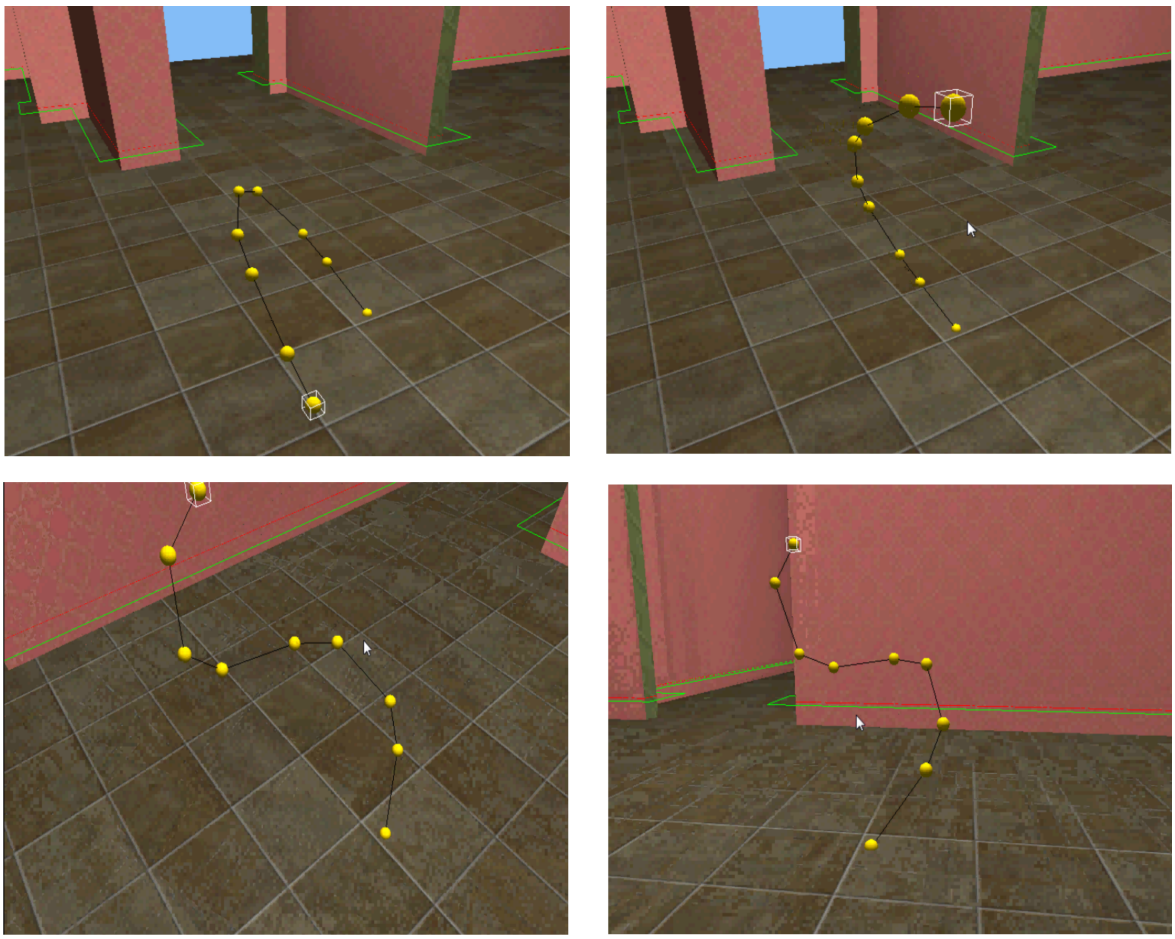


FIGURE 3.4.1: Mouvement de la chaine articulée

Par la suite, nous avons integré la méthode CCD à cette chaine pour pouvoir la comparer avec la méthode FABRIK. Nous avons ainsi établi une comparaison en terme de nombre d'itérations, de temps de calcul et de nombre d'images par seconde présentée dans le tableau suivant :

Méthode	CCD	FABRIK
Nombre d'itérations	17.6	13.2
Temps de calcul (Secondes)	1.2	0.3
Images par secondes	16	55

TABLE 3.2: Résultats de la comparaison

Dans notre cas, on peut remarquer que la méthode FABRIK s'exécute en moins d'itérations que la méthode CCD et que le temps de calcul est largement avantageux pour FABRIK. Notons que ce temps de calcul est pour les 100 exécutions.

Pour le nombre d'images par seconde, le résultat pour FABRIK est supérieur à celui de CCD. Ce paramètre précise combien de fois l'algorithme a été appelé en une exécution : il définit la fluidité du mouvement et la rapidité de de l'algorithme.

3.4.2 Implémentation sur l'avatar

Nous avons intégré la méthode FABRIK sur l'avatar de HandiAccess et exécuté l'application afin qu'il atteigne une cible. La même opération a été appliqué avec la méthode CCD. Le tout nous a permis d'obtenir les postures suivantes :

FIGURE 3.4.2: Implémentation sur l'avatar

Sur cette figure, nous constatons une certaine similitude dans les postures finales de l'avatar découlant des deux méthodes, à une différence près, celle du poignet. Ce dernier a une position plus correcte sur FABRIK, tout en sachant que, pour quasiment la même posture, FABRIK présente de meilleurs résultats comme cités dans le tableau (3.2).

3.5 Conclusion

Nous avons entamé ce chapitre par une brève description de l'application HandiAccess et des outils de développement utilisés dans sa conception, puis, nous avons implémenté la méthode FABRIK dans sa version sans contraintes et nous avons pu la comparer avec la méthode CCD déjà présente dans l'application. De là, nous avons constaté, après comparaison entre les deux méthodes, les avantages de la méthode FABRIK en terme de rapidité, fluidité, et temps de calcul.

Conclusion générale

Les travaux de recherche présentés dans ce mémoire avaient pour but de contribuer à améliorer le projet HandiAccess : projet capable d'évaluer et de simuler l'accessibilité de l'environnement pour les personnes à mobilité réduite. Cette contribution s'est caractérisée par l'implémentation de la méthode itérative FABRIK et sa comparaison avec les résultats de la méthode CCD.

Grâce à l'algorithme FABRIK, nous avons pu constater que l'application :

- Produit des positions visuellement lisses,
- Atteint la position désirée avec un coût en termes de temps de calcul très bas,
- Exige en moyenne moins d'itérations que celles données par la méthode CCD.
- Et pour le nombre d'images par seconde, le résultat pour FABRIK est supérieur à celui de CCD.

Les résultats obtenus à partir de notre application démontrent la capacité de la méthode FABRIK à résoudre de manière optimale les problèmes de contrôle cinématique du mouvement.

Dans un souci d'amélioration, et pour de futurs travaux de recherche, nous proposons l'implémentation de FABRIK avec la prise en compte des contraintes et/ou la prise en compte de chaines multiples.

Annexes

Annexe A

Généralités mathématiques

Dans cette section, nous présentons quelques notions qui nous sont utiles tout au long de notre travail.

A.1 Degrés de liberté

On entend parler de cette notion de degrés de liberté dans plusieurs domaines tels en physique, en chimie, en statistique et en mécanique.

1. En physique et en chimie : Le degré de libérté indique la possibilité pour un système d'évoluer dans une direction non contrainte.
2. En statistique : Il désigne le nombre de variables aléatoires qui ne peuvent être déterminées ou fixées par une équation (notamment les équations des tests statistiques).
3. En mécanique : Le degré de liberté est la possibilité de se déplacer et de bouger comme on veut dans l'espace.

D'une façon générale il peut être défini comme étant le nombre de variable indépendante qui permettent de décrire un objet

Degrés de liberté en robotique

La position d'un solide indéformable isolé dans l'espace ou par rapport à un autre référent est définie par six paramètres (trois coordonnées et trois angles). Alors, un solide libre possède six degrés de liberté ce qui veut dire 6 mouvements sont considérés : trois translations et trois rotations de directions indépendantes. Si on s'oriente dans l'espace (à 3 dimensions) à l'aide d'un repère orthonormé (O, x, y, z), Les six degrés de liberté s'expriment comme suit :

1. En translation
   - Avant - arrière (x)

- Droite- gauche (y)
- Haut - bas (z)

2. En rotation
   - Basculer d'avant en arrière (Tangage)
   - Basculer de droite à gauche (Roulis)
   - Pivoter comme les aiguilles d'une montre (Lacet)

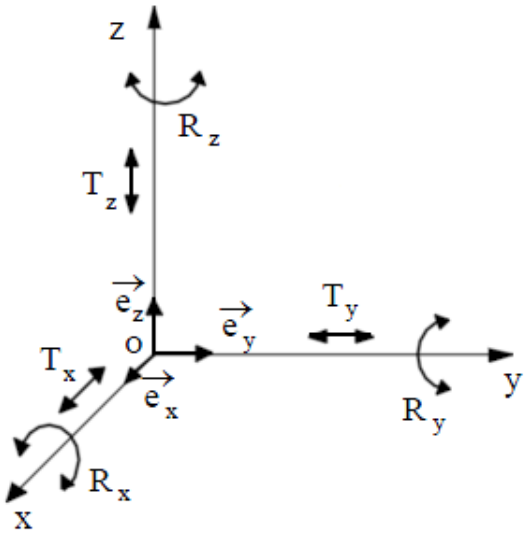


FIGURE A.1.1: Les six degrés de liberté[32]

Modèle articulé du corps humain

Toute la partie supérieure du corps humain peut être modélisée sous la forme d'une chaîne articulée. Chaque articulation possède un nombre de degré de liberté spécifique. La figure (A.1.2) suivante décrit la répartition des degrés de liberté [12].

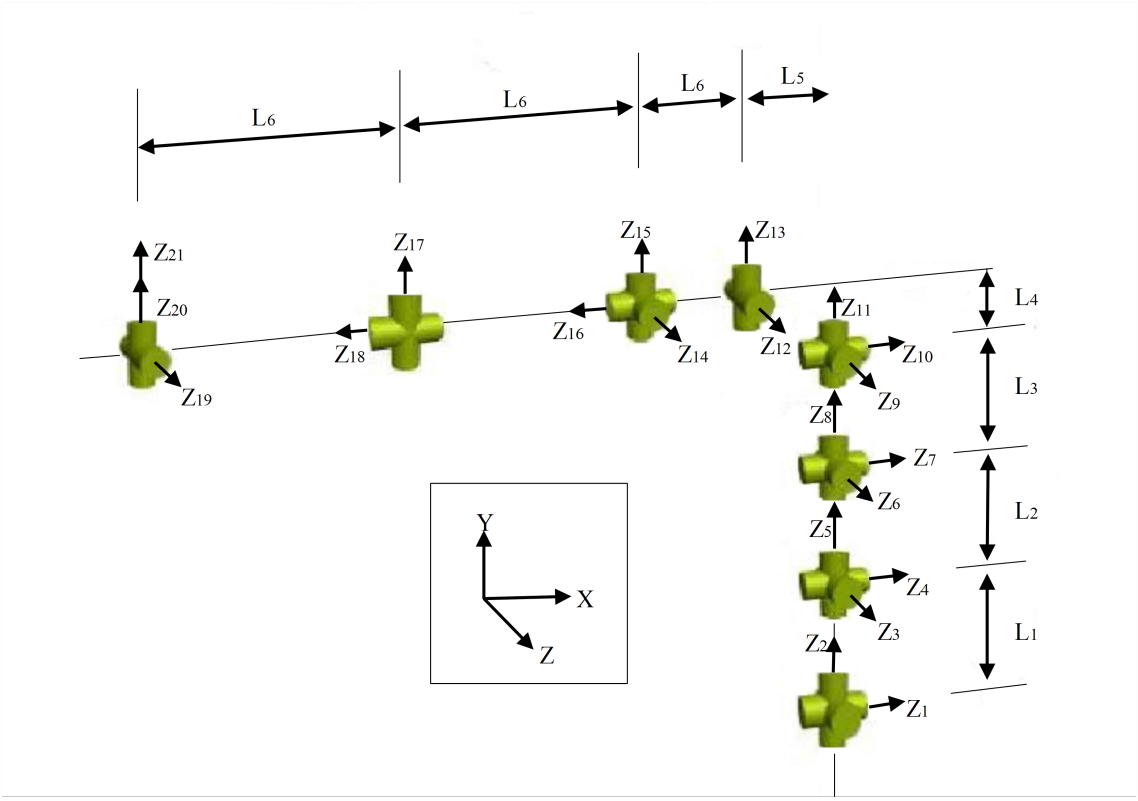


FIGURE A.1.2: Modèle à 21 degrés de liberté

A.2 Le produit vectoriel

Définition

Le produit vectoriel est une notion de mathématiques, et plus précisément de géométrie. C'est l'opération vectorielle effectuée dans les espaces orientés de dimension trois. Dans une base orthonormée, Le produit vectoriel s'écrit [33] :

$$\overrightarrow{u} \wedge \overrightarrow{v} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \wedge \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} + \begin{vmatrix} x_2 & y_2 \\ x_3 & y_3 \end{vmatrix} \\ - \begin{vmatrix} x_1 & y_1 \\ x_3 & y_3 \end{vmatrix} \\ + \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} \end{pmatrix} = \begin{pmatrix} x_2.y_3 - x_3.y_2 \\ x_3.y_1 - x_1.y_3 \\ x_1.y_2 - x_2.y_1 \end{pmatrix}$$

Comme il peut aussi être calculé de la façon suivante :

$$\| \overrightarrow{u} \wedge \overrightarrow{v} \| = \| \overrightarrow{u} \| \, \| \overrightarrow{v} \| \, |sin(\overrightarrow{u}, \overrightarrow{v})|$$

Les différentes notations

Soit u et v deux vecteurs, alors leur produit vectoriel est noté comme suit :

1. $u \wedge v$ en France où le symbole $\wedge$ veut dire vectoriel.
2. $u \times v$ dans la littérature anglophone, au Canada francophone et en Suisse.
3. $[u, v]$ notation utilisant les crochets de Lie .

Annexe B

Code source de FABRIK

```
void CChaine : :ik(float tarx,float tary,float tarz)
{
Ogre : :Vector3* t=new Ogre : :Vector3( tarx, tary, tarz) ;
vector<float> d ;
float delta=1 ;
vector<Ogre : :Vector3*> : :iterator itart ;
vector<float> : :iterator itd ;
vector<Ogre : :Vector3*> : :reverse_iterator ritart ;
vector<float> : :reverse_iterator ritd ;
float sum_di=0 ;
//Calcul des di
for(itart =posArt.begin()+1 ; itart< posArt.end() ; itart++)
{
float di=(*itart)->distance(**(itart-1) ) ;
sum_di+=di ;
d.push_back(di) ;
}
// Calcul de dist
float dist=t->distance(*posArt[0]) ;
// Check whether the target is within reach
if(dist>sum_di)
{
// Pas de solution
sum_di ;
```

```
// Find the distance ri between the target t and the joint position pi
for(itart =posArt.begin(),itd=d.begin() ; itart< (posArt.end()) ; itart++,itd++)
{
if((*itart)==posArt.back())break ;
float lambda=(*itd)/ri ;
(*(*(itart+1)))=(1-lambda)*(*(*(itart)))+lambda*(*t) ;
}
}
else
{
//il y a une solution
// The target is reachable ; thus, set as b the initial position of the joint p1
Ogre : :Vector3* b=new Ogre : :Vector3(0.0) ;
*b=(**posArt.begin()) ;
// Check whether the distance between the end effector pn and the target t is greater
than a tolerance.
float difA=(*posArt.back()).distance(*t) ;
while(difA>delta)
{
// STAGE 1 : FORWARD REACHING
// Set the end effector pn as target t
(*posArt.back())=*t ;
for(ritart =posArt.rbegin()+1,ritd=d.rbegin() ; ritart< (posArt.rend()) ; ++ri-
tart,++ritd)
{
// Find the distance ri between the new joint position pi+1 and the joint pi
float r=(*(*(ritart-1))).distance((*(*(ritart)))) ;
float lambda=(*ritd)/r ;
// Find the new joint positions pi.
(*(*(ritart)))=(1-lambda)*(*(*(ritart-1)))+lambda*(*(*(ritart))) ;
}
// STAGE 2 : BACKWARD REACHING
// Set the root p1 its initial position.
(**posArt.begin())=*b ;
for(itart =posArt.begin(),itd=d.begin() ;
itart< (posArt.end()) ; itart++,itd++)
{
if((*itart)==posArt.back())break ;
```

```
// Find the distance ri between the new joint position pi and the joint pi+1
float ri=(*itart)->distance(*(*(itart+1)));
float lambda=(*itd)/ri;
// Find the new joint positions pi.
(*(*(itart+1)))=(1-lambda)*(*(*(itart)))+lambda*(*(*(itart+1)));
}/**/
difA=(*posArt.back()).distance(*t);
}
}
}
```

Bibliographie

[1] http://lequotidienalgerie.org/2011/12/01/la-personne-handicapee-entre-legislation-et-droit-a-la-dignite/

[2] Philippe FUCHS, Guillaume MOREAU, Alain BERTHOZ, Jean-Louis VERCHER, "L'homme et l'environnement virtuel", Le traité de la réalité virtuelle - Volume 1, Ecole des MINES de Paris, 2006.

[3] Philippe FUCHS, Guillaume MOREAU, Jean-Marie BURKHARDT, Sabine COQUILLART, "L'interfaçage, l'immersion et l'interaction en environnement virtuel", Le traité de la réalité virtuelle - Volume 2, Ecole des MINES de Paris, 2006.

[4] http://www.ceremh.org/accessibilite/recherche-et-innovation-47/accessim/

[5] http://rnt.over-blog.com/article-11129274.html

[6] Philippe VEZIN, "La sécurité des transports,réduire les risques de blessures : de l'expérimentation biomécanique aux modèles humains", INRETS /université Claude Bernard Lyon 1.

[7] http://www.nvidia.fr/object/virtual-automotive-design-fr.html

[8] http://www.experts.renault.com/kemeny/phds/nicolas_filliard/

[9] Bernard BAYLE,"Introduction à la Robotique", IUP Technologies Avancées des Sciences du Vivant, Université Louis Pasteur de Strasbourg.

[10] André PREUMONT, "Théorie Générale des Systèmes Articulés", Laboratoire des Structures Actives Service des Constructions Mécaniques et Robotique de l'université Libre de Bruxelles, 2001.

[11] http://entrainement-sportif.fr/anatomie-humaine.htm, Anatomie- épaule - Omoplate, humérus , clavicule, les os de l'épaule. .

[12] Bilal BOUALEM, "Évaluation et simulation de l'accessibilité d'un lieu de vie", Projet de Fin d'Etude pour obtention du diplôme de master, Université ABB, 2010.

[13] `http://www.clinique-du-dos.com/dico/les-vertebres.php`, Les vertèbres

[14] `http://www.futura-sciences.com/fr/doc/t/medecine-1/d/arthrose-origine-et-traitement-de-larthrose_888/c3/221/p8/`

[15] `http://www.medecine-et-sante.com/anatomie/main.html`

[16] Andreas ARISTIDOU, Joan LASENBAY, "FABRIK : A fast, iterative solver for the Inverse Kinematics problem", Graphical Models 73 (2011) 243–260, 2011.

[17] Bernard BAYLE,"Introduction à la Robotique", IUP Technologies Avancées des Sciences du Vivant, Université Louis Pasteur de Strasbourg.

[18] Rachid ALOUANI, "Analyse et Commande d'une structure articulée", Institut Supérieur de l'Automatique et Electronique, Université Paul Verlaine – Metz, 2009

[19] Vincent PADOIS, Wael BACHTA, "Commande des Systèmes Robotiques", Université Pierre et Marie Curie Institut des Systèmes Intelligents et de Robotique.

[20] Donatien NGANGA-KOUYA, Francis A.OKOU, Méthodologie hybride de conception de la commande force/position des robots manipulateurs, Afrique SCIENCE 05(3) 111 - 127, 2009.

[21] Vincent BONNAFOUS, Eric MENOU, Jean Pierre JESSEL, René CAUBET, "Une plate-forme d'animation utilisant conjointement plusieurs méthodes de contrôle du mouvement", Université Paul Sabatier.

[22] Matthieu AUBRY, "Modélisation et apprentissage de synergies pour le contrôle du mouvement de personnages virtuels", Université Européenne de Bretagne, 2010.

[23] L.Flückiger, "Interface pour le pilotage et l'analyse des robots basée sur générateur de cinématique", Thèse de doctorat des sciences techniques, école Polytechnique fédérale de Lausanne, Suisse, 1998.

[24] M.H.Lavoie, "Solution par optimisation numérique du problème géométrique inverse de manipulateur mobile sériels redondants pour des opérations de transport dans un espace encombré", Maîtrise des sciences appliquées, Faculté d'ingénierie, Université de Moncton, Canada, Octobre 2001.

[25] Isma AKLI, "Elaboration d'une strategie de coordination de mouvements pour un manipulateur mobile redondant", mémoire présenté pour l'obtention du diplôme de magister, Université U.S.T.H.B, 2007

[26] Nicolas COURTY, Elise ARNAUD, "Inverse Kinematics using Sequential Monte Carlo Methods", Universite Joseph Fourier, INRIA Rhône-Alpes, LJK, Grenoble, France, 2008.

[27] WANG Li-Chun Tommy, CHEN Chih Cheng. "A combined optimization method for solving the inverse kinematics problem of mechnical manipulators", IEEE Transactions on Robotics And Automation, 1991.

[28] Ludovic HOYET, "Adaptation dynamique de mouvements de sauts pour des personnages de synthèse", Rapport de stage de master recherche, I R I S A

[29] `http://www.robertnz.net/nm_intro.htm`

[30] Richard GOURDEAU, "ROBOOP A Roboticzs Object Oriented Package in C++", Département de génie électrique Ecole Polytechnique de Montréal, 2007.

[31] Julien CHICHIGNOUD, "Découvrez Ogre 3D", siteduzero.com, 2012.

[32] Catherine POTEL, Philippe GATIGNOL, "Laisons mécaniques - modélisation", Université du Maine.

[33] Julien Dompierre, "Produit vectoriel Algèbre linéaire I", Département de mathématiques et d'informatique Université Laurentienne, 2011.