REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SURERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

Département d'Automatique

Projet de fin d'études

En vue de l'obtention du diplôme d'ingénieur d'état en Automatique

Thème :

Conception et implémentation de programme de commande pour la plate-forme robotique B21r sous ROS

Réalisé et présenté par:

Mahtout Imane

Yahoui Anissa

Proposé et dirigé par :

Pr H.Chekireb

Ouadah Nourdine(cdta)

Juin 2015

Ecole Nationale Polytechnique, 10, AV. Hassen Badi, El-Harrach, Algérie

ملخص :

تتمثل هذه الدراسة في تصميم و تنفيذ قوانين التحكم باستعمال البنية البرمجية ROS ، قادرة على التحكم بالروبوت B21r و جعله مستقل بذاته. سنركز في الجزء الأول على التمكن من البرامج الضمنية للروبوت و التي تعمل على إدارة المعلومات الآتية من جهاز الاستشعار RGB-D و المراجع المرسلة للمشغلات. يتمثل الجزء الثاني في تصميم عقدة ROS لتنفيذ انواع مختلفة من التحكم البصري ثنائي و ثلاثي الأبعاد. في الأخير، سنتحقق من صحة هذا العمل بدمج عقد جديدة في الهندسة الاجمالية و عن طريق التجربة بمهام آلية متقدمة.

الكلمات المفتاحية :

قوانين التحكم, الروبوت B21r, ROS عقدة , جهاز الاستشعار RGB-D , التحكم البصري ثنائي و ثلاثي الأبعاد

Résumé :

Le travail consiste à concevoir puis implémenter des lois de commande dans une architecture logicielle sous ROS, capable de contrôler le robot B21r, et cela dans le but de le rendre autonome. Dans la première partie, on s'intéressera a la maitrise des programmes embarqués sur le robot et qui servent à gérer les données du capteur RGB-D et les références envoyés aux actionneurs. La deuxième partie consiste à concevoir un nœud ROS pour implémenter différentes types d'asservissement visuel 2D et 3D. Au final, la validation de ce travail se fera par l'intégration de nouveaux nœuds dans l'architecture globale et les tests expérimentaux pour des tâches robotique évoluées.

Mots Clés :

Lois de commande, robot B21r, nœud ROS ,Asservissement visuel 2D et 3D, capteur RGB-D .

Abstract:

This work consists of conception and implementation of control laws under ROS software architecture, capable of controlling the robot B21r, in order to make it autonomous. In the first part, we will focus on mastering the embedded programs managing the data coming from the RGB-D sensor and the references sent to the actuators. The second part consists of conception of an ROS node in order to implement different types of 2D and 3D visual control. Finally, the work will be validated by the integration of new nodes into the global architecture and the experimental test for advanced robotic tasks.

Keywords:

Control laws , robot B21r , ROS node , 2D and 3D visual control , RGB-D sensor.

Dédicace

À la femme la plus chère au monde, qui a fait de moi ce que je suis aujourd'hui

Ma mère

À mon idole qui m'a appris à être autonome et aimer le travail

Mon père

À mes chères sœurs Nawel Et Nesrine,

A ma petite cousine maline Malék,

A tous les membres de ma famille : oncles, tantes, cousins et cousines,

À toute la Famille Mahtout et Melakhi

À Ma chère binôme qui m'a bien supportée: Nissa

À tous mes enseignants,

À tous ceux que j'aime et qui m'aiment.

Mahtout Imane

Dédicace

À l'homme le plus cher au monde mon père,

À mon trésor ma mère,

À mes très chers grands-parents,

À mon cher frère et ma chère sœur,

A ma petite cousine adorée Lyna,

A tous les membres de ma famille : oncles, tantes, cousins et cousines,

À toute la Famille Yahoui et Boudjemai

À tous mes enseignants,

À Ma chère binôme et ma meilleure amie : Imane ,

À tous ceux que j'aime et qui m'aiment.

Yahoui Anissa

Remerciements

Louange à Allah le tout puissant qui nous a accordé le droit chemin, le savoir et l'opportunité de poursuivre nos études et la force pour réaliser ce modeste travail.

Au terme de notre cycle d'études, il nous paraît opportun de nous acquitter d'un devoir noble, celui de remercier tous ceux qui ont contribué par leur assistance tant morale que physique à notre cursus universitaire et à la réalisation de ce mémoire .

Nous tenons à exprimer nos vifs remerciements à nos encadreurs :

***Mr El-Hachemi Chekireb** professeur à l'école nationale polytechnique*

***Mr Noureddine Ouadah** docteur et chercheur au CDTA,*

Pour leurs disponibilités, aides et bonnes humeurs durant toutes les étapes de ce projet. Leurs dévouements, conseils scientifiques et suivis, nous ont permis de mener notre travail à terme.

Nous remercions les membres du jury d'avoir bien voulu étudier ce travail et participer à la commission d'examen afin de le juger.

Nous souhaitons aussi remercier tous les enseignants de l'école nationale polytechnique en particulier ceux du département d'automatique pour l'enseignement qui nous ont transmis.

Que tous ceux qui ont contribués de près ou de loin à la réalisation de ce modeste travail trouvent ici l'expression de notre sincère gratitude

Merci infiniment à tous.

Sommaire

Table des figures

Notation

$\boldsymbol{2D}$: Deux dimensions.

$\boldsymbol{3D}$: Trois dimensions.

$\boldsymbol{e}$: Fonction de tache.

$\boldsymbol{t}$: Le temps(s).

$\boldsymbol{s}$: Indices visuels .

$\boldsymbol{s^*}$: Indices visuels désirés.

$\boldsymbol{C}$: Matrice de combinaison.

$\boldsymbol{L}$: Matrice d'interaction 2D.

$\boldsymbol{L_r}$: Matrice d'interaction réduite 2D.

$\boldsymbol{\mathcal{L}}$ **:** Matrice d'interaction 3D.

$\boldsymbol{\mathcal{L}_r}$ **:** Matrice d'interaction réduite 3D.

$\boldsymbol{z}$: Distance entre la Kinect et la cible appelée profondeur (m).

$\boldsymbol{z^*}$: Profondeur désirée(m).

$\boldsymbol{\hat{z}}$: Profondeur estimée(m).

$\boldsymbol{r}$: Situation de la Kinect par rapport à la scène.

$\mathbf{R_O}$: Repère lié à la scène.

$\mathbf{R_M}$:Repére lié à la base mobile.

$\mathbf{R_k}$: Repère lié à la Kinect.

$\boldsymbol{l}$: Abscisse curviligne de la base mobile(m).

$\boldsymbol{\vartheta}$: Orientation de la base mobile (rad).

$\boldsymbol{t_x}$: Abscisse de l'origine de la Kinect dans$\mathbf{R_M}$

$\boldsymbol{t_y}$: Ordonné de l'origine de la Kinect dans$\mathbf{R_M}$ **.**

$\mathbf{T_{K/R_o}}$:Torseur cinématique de la Kinect.

$\mathbf{V_{K/R_o}}$ **:** vitesse de translation de la Kinect (m/s).

$\boldsymbol{\Omega_{K/R_o}}$ **:** vitesse de rotation de la Kinect (rad/s).

$\mathbf{V}\boldsymbol{X_K}$:vitesse de translation de la Kinect suivant l'axe $\boldsymbol{X_K}$(m/s).

$\mathbf{V}\boldsymbol{Y_K}$:vitesse de translation de la Kinect suivant l'axe $\boldsymbol{Y_K}$(m/s).

$\mathbf{V}\boldsymbol{Z_K}$ **:** vitesse de translation de la Kinect suivant l'axe $\boldsymbol{Z_K}$(m/s).

$\boldsymbol{\Omega X_K}$:vitesse de rotation de la Kinect autour de l'axe $\boldsymbol{X_K}$(rad/s).$\boldsymbol{\Omega Y_K}$ **:** vitesse de rotation de la Kinect autour de l'axe $\boldsymbol{Y_K}$(rad/s).

$\boldsymbol{\Omega Z_K}$:vitesse de rotation de la Kinect autour de l'axe $\boldsymbol{Z_K}$(rad/s).

$\boldsymbol{q}$: Vecteur de configuration du robot.

$\boldsymbol{\dot{q}}$: Vecteur commande du robot.

$\boldsymbol{J}$:Jacobien du robot.

$\boldsymbol{J_r}$: Jacobien réduit du robot.

$\boldsymbol{f}$:distance focale.

$\boldsymbol{\lambda}$**:** gain de l'asservissement**.**

$(\boldsymbol{u}, \boldsymbol{v})$:Coordonnées de la cible dans l'image en pixels (pix).

$(\boldsymbol{x'}, \boldsymbol{y'})$: Coordonnées de la cible dans l'image.

$(\boldsymbol{k_u}, \boldsymbol{k_v})$ **:** Facteur d'échelle vertical et celui horizontal dans l'image

RGB-D: Red green blue depth.

IR: Infra-red.

INTRODUCTION GENERALE

De nos jours, les tâches robotisées se présentent comme une alternative sérieuse à certaines activités humaines surtout pour les tâches répétitives et pénibles. Par ailleurs, les robots en général sont classés en deux grandes familles : les robots manipulateurs industriels et les robots mobiles. Les manipulateurs industriels sont surtout destinés à opérer en milieu l'industriel. Par contre, les robots mobiles (à roues, marcheurs, volants, …) sont le plus souvent destinés à des applications spécifiques (exploration, inspection, manipulation d'objets, accompagnement, …).

Le déploiement des robots mobiles en environnement humain privatif mais aussi public répond à un enjeu sociétal majeur. Il s'agit à terme de voir ces robots interagir de façon naturelle et effective avec les humains. En effet, l'interaction ne peut se faire sans équiper le robot par des capteurs assurant la perception de son environnement. Un exemple, le suivi de personne est une tâche basique qui permet d'initier et maintenir cette interaction.

Le sujet à traiter, pour notre mémoire, rentre dans cet important contexte. Ainsi, le problème consiste à faire naviguer un robot mobile à roues dans un milieu intérieur, sur la base des informations d'un capteur RGB-D. Le traitement des informations issues de ce capteur permet l'identification et le suivi des personnes, d'où le choix d'implémenter des lois de commande de type asservissement visuel 2D et 3D, afin de contribuer à résoudre le problème cité.

Le travail effectué dans le cadre de ce mémoire est présenté dans quatre (04) chapitres :

Au chapitre I, nous posons le problème général de l'asservissement visuel dans son aspect commande référencée capteurs, ainsi que le formalisme de la fonction de tâche et les différentes techniques de l'asservissement visuel.

Au chapitre II, nous présentons la plateforme expérimentale, le capteur de vision utilisé ainsi que les notions de géométrie projective. Nous détaillerons par la suite les lois de commande utilisées. La première commande est synthétisée en considérant la tâche comme un problème de régulation dans l'image, puis en second lieu la question est vue comme une tâche de positionnement de la cible dans l'espace capteur selon deux variantes :

- en se basant sur la mesure 3D d'un référentiel lié à la cible ;
- en se basant sur des points 3D de la cible.

Au chapitre III, nous présentons, l'architecture logicielle de notre système robotique : le robot B21r (mis à notre disposition au CDTA).

Au Chapitre IV, nous procédons à une expérimentation des commandes visuelles proposées afin d'évaluer leurs performances. Nous présentons tout d'abord les résultats de l'asservissement 2D, selon les différents choix de calcul de la matrice d'interaction. L'objectif étant de comparer les performances des trois variantes existantes dans la littérature. Dans la deuxième partie, la même méthodologie est adoptée pour les deux commandes visuelles3D.

Nous terminerons ce manuscrit pas une conclusion générale, et des perspectives décrivant les suites à donner à cette étude. Il est à noter que notre travail rentre dans le cadre d'un projet sur le B21r au CDTA impliquant une équipe scientifique de la division robotique et productique ou chacun se charge d'une tâche spécifique pour aboutir au final aux objectifs ciblés par l'équipe.

CHAPITRE 1

Généralités sur l'asservissement visuel

1. Introduction

Dans ce chapitre, nous présentons le problème général de l'asservissement visuel dans son aspect commande référencée capteurs et nous passons en revue les grandes approches et méthodes de résolution proposées en littérature de ce fait ce chapitre est divisé en deux parties :

- La première partie concerne la commande par vision et les formalismes associés.
- La deuxième partie est dédiée à la présentation des différentes techniques de l'asservissement visuel et leurs avantages et inconvénients.

2. Commande par vision

Comme dans toute boucle d'asservissement, un bloc de commande est dédié au calcul de la commande, nécessaire à l'exécution d'une tache, en exploitant les informations fournies par les capteurs de mesure.

Dans le cas de l'utilisation d'un capteur de vision, la commande peut être du type référencée capteur où la référence correspond à une image bien spécifiée que doit fournir le capteur de vision. Cette commande a été développée durant les années 90 ; c'est une méthodologie complète de programmation des tâches robotiques allant de la spécification jusqu'à la synthèse des lois de commande. Cette méthode peut être appliquée à une grande variété de capteurs tels que les caméras, les capteurs de force et les capteurs télémétriques.

2.1 Méthodologie de la commande référencée capteur

2.1.1 Signaux de mesure

En situation de la commande référencée capteur, le capteur est généralement monté rigidement sur une des liaisons du robot auquel est associé un repère **C**. Ce capteur fournit un signal **S** relatif à la perception d'un objet cible dans son environnement.

Nous admettons, comme hypothèse de base, que le signal **S** ne dépend que de la configuration relative **r** (position et orientation) entre le repère du capteur et un repère de référence **R** lié à la cible (Fig.1-1). Cela signifie que si le capteur est statique par rapport à la cible, alors le signal **S** est stationnaire.

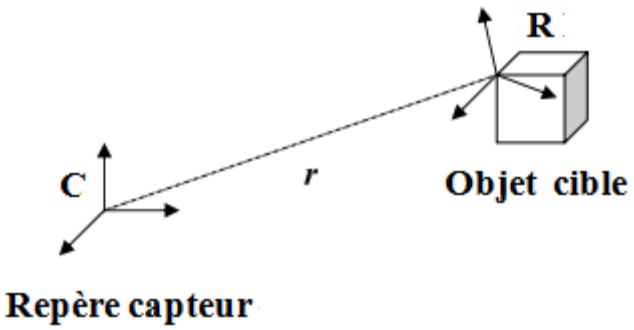


Fig.1.1 Espace de configuration d'un capteur

2.1.2 Approche de la fonction de tâche

Le signal de perception provenant du capteur étant défini, nous avons besoin de formuler la tâche robotique en termes d'une régulation dans le but de déterminer par la suite la loi de commande assurant la réalisation de cette tâche. Ceci justifie le recours au formalisme de la fonction de tâche.

Ce formalisme a été initialement développé par **Samson et al** mais son utilisation était limitée à la commande des bras manipulateurs. Il fut ensuite étendu aux cas des robots mobiles non-holonomes par **Pissard** (oudah, 2012).

Le formalisme de la fonction de tâche est basée sur la définition d'une fonction **e(q,t)** où q est le vecteur de configuration du robot et l'objectif de la tâche sera satisfait si **e(q,t)**converge vers zéro. Cette fonction peut être exprimée en termes de données proprioceptives ou extéroceptives selon l'application considérée et les capteurs disponibles sur le robot.

Les fonctions de tâche classiques sont définies telles que:

- **e (q,t)** = **q(t) – q*(t)** où **q*(t)** est une trajectoire désirée dans l'espace des configurations du robot.
- **e (q,t)** = **r(s (q,t)) - r*(s*(t))** où **r*(s*(t))** est une trajectoire désirée dans l'espace cartésien estimée à partir des informations visuelles **s*(t)**.
- **e (q,t)= s (q, t) - s*(t)** où **s*(t)** est une trajectoire désirée définie dans l'espace du capteur.

Cependant, la fonction de tâche pratiquement utilisée est de la forme : **e(q,t)= C(s (q,t) - s*(t))** où la matrice **C** est une matrice de combinaison. Cette dernière permet de prendre en compte le fait que le degré de liberté du robot est inférieure au nombre d'indices visuels. Il est important de noter que l'introduction de cette matrice peut perturber la convergence des indices visuels car la convergence de **e(q,t)** vers zéros assure uniquement la convergence de **Cs(q, t)** vers **Cs*(t)** et non celle de **s (q, t)** vers **s*(t).**

2. 2 Asservissement visuel cinématique

Dans ce paragraphe, nous définissons le vecteur de la configuration du robot qui nous permettra par la suite de commander le robot dans le but d'annuler la fonction de tache.

En asservissement visuel, les lois de commandes sont de la forme cinématique où les vitesses articulaires sont calculées périodiquement. Celles-ci servent comme consignes aux contrôleurs bas niveau du robot.

La cadence de rafraîchissement des consignes est imposée par la fréquence d'acquisition des images, la durée des traitements nécessaires à l'extraction de l'information visuelle et le calcul de la loi de commande.

3. Techniques de l'asservissement visuel

Les techniques de l'asservissement visuel sont des techniques qui permettent d'intégrer les mesures issues d'une ou plusieurs caméras dans l'asservissement d'une boucle afin de maitriser le mouvement d'un robot et ceci dans le but de réaliser différentes taches comme le positionnement du robot dans son environnement, la poursuite d'un objet mobile …etc.

Il existe plusieurs approches d'asservissement visuel qui se distinguent par le type de données fournies par les capteurs (par exemple : données 2D ou 3D) et par le type de la boucle d'asservissement.

Dans ce qui suit, nous présentons une classification des principales techniques.

3.1 Asservissement visuel 2D

On parle d'asservissement visuel **2D** lorsque l'information visuelle est définie dans le plan image d'où sont extraites les primitives de l'image. Les primitives sont des formes géométriques élémentaires (point, segment de droite, portion d'ellipse…) associées à l'image. Ceci permet de modéliser la projection de la cible dans le plan image.

Une fois les indices visuels sélectionnés, l'asservissement **2D** consiste à contrôler le mouvement du robot de façon à déplacer les primitives observées dans l'image, de la position courante $\boldsymbol{s}$ vers la position désirée $\boldsymbol{s}^*$. Ceci est obtenu**,** en synthétisant une commande assurant la convergence vers zéro d'une fonction de tâche définie dans l'image qui doit permettre la convergence de la valeur courante des primitives visuelles $\boldsymbol{s}$ vers leurs valeurs désirées$\boldsymbol{s}^*$. On choisit généralement **N** primitives de type point avec **N≥4** (**Fig. 1-2**).

Pour pouvoir synthétiser la commande en asservissement visuel 2D, il est nécessaire d'établir le lien entre le torseur cinématique du repère de la caméra et les vitesses de déplacement des primitives dans l'image.

Le pionnier en matière d'asservissement visuel 2D fut Weiss et al. Il étudia ce lien et définit la matrice d'interaction notée $\boldsymbol{L}(\boldsymbol{s},\boldsymbol{z})$ ou $\boldsymbol{s}$ représentent les coordonnées des primitives visuelles et $\boldsymbol{z}$ représente la profondeur entre la caméra et la cible (Chaumette, 2002).

La matrice $\boldsymbol{L}(\boldsymbol{s},\boldsymbol{z})$est exprimée par la différentielle de $\boldsymbol{s}$permettant de relier les variations des indices visuels à la variation de la situation de la caméra par rapport à la scène.

Nous obtenons par dérivation :

$$\dot{\boldsymbol{s}} = \frac{\boldsymbol{ds}}{\boldsymbol{dr}}\dot{\boldsymbol{r}} = \boldsymbol{L}(\boldsymbol{s},\boldsymbol{z})\dot{\boldsymbol{r}}$$

Ou $\boldsymbol{r}(\boldsymbol{t})$ représente la situation de la caméra par rapport à la scène.

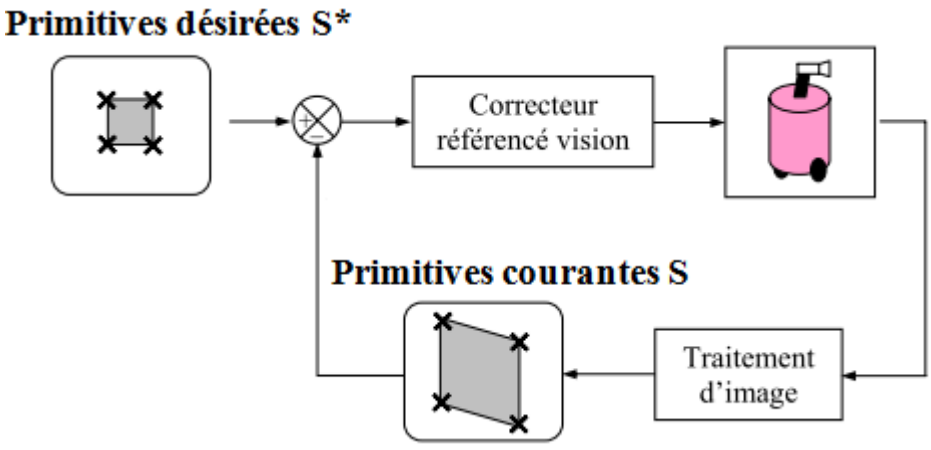


Fig1.2 Asservissement visuel 2D

Cette matrice est introduite dans une boucle d'asservissement selon deux approches dépendant du calcul en ligne ou hors ligne de$\boldsymbol{L}(\boldsymbol{s},\boldsymbol{z})$.

- **Calcul de $L(s,z)$en hors ligne**

  Dans ce cas la matrice est calculée une fois avant que la loi de commande soit implémentée car on met $\boldsymbol{L(s,z)} = \boldsymbol{L(s^*,z^*)}$ou $\boldsymbol{s^*}$et $\boldsymbol{z^*}$représentent respectivement la valeur des indices visuels et la profondeur désirés. Cette approche a été adoptée pour alléger les calculs dans la boucle en préservant la convergence mais celle-ci n'est pas garantie si la position initiale de la cible est assez loin de celle désirée.

- **Calcul de $L(s,z)$ en ligne**

Contrairement au cas précédent, il faut, à chaque itération, actualiser la valeur de la matrice $\boldsymbol{L(s,z) = L(s(t),z(t))}$ qui dépend non seulement de la valeur des indices visuels dans l'image mais aussi de la profondeur. Celle-ci est soit fournie par un autre capteur comme la caméra **RGB_D** ou soit estimée à partir du modèle **3D** de la cible ; nous obtenons donc$\boldsymbol{L(s(t),\hat{z})}$**.**

Ces deux approches présentent des avantages et inconvénients vis-à-vis de la convergence des indices visuels.

- Le calcule de $\boldsymbol{L(s,z)}$ en ligne contraint fortement la trajectoire que doit suivre les indices visuels dans l'image pour atteindre les valeurs désirées et c'est une trajectoire contenue dans l'image ce qui est souhaitable (**Fig.1.3**). Cependant, ceci peut conduire à des mouvements inadmissibles de la caméra (singularité) ce qui peut engendrer des minimas locaux de l'erreur dans l'image.

- Le calcule hors ligne ne contraint pas la trajectoire à rester dans l'image (**Fig.1.3**) de plus ; la camera est pratiquement moins sujette aux mouvements inadmissibles et aux minimas locaux. Ceci est obtenu dans la condition où la position initiale de la caméra est peu éloignée de la position désirée sinon la convergence est perdue.

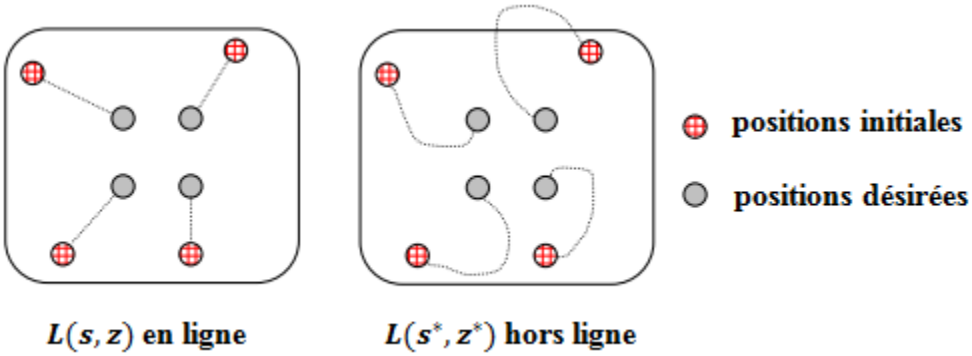


Fig.1.3 Trajectoire des indices visuels dans l'image

- Une troisième approche, étudiée récemment, consiste en une combinaison linéaire des deux précédentes approches d'où $\boldsymbol{L(s,z) = \left[L\left(s(t),z(t)\right) + L(s^*,z^*)\right]/2.}$ L'expérience a montré que cette dernière méthode conduit à de meilleurs résultats. Cependant, les trois approches présentent l'inconvénient de se bloquer dans des situations de singularité ou de mouvements inadéquats de la caméra.

3.2 Asservissement visuel 3D

Dès 1979 et pour améliorer la précision de la tache de positionnement des robots [Tamtsia, 2013], Shirai et Hill ont exploité les techniques de l'asservissement visuel 3D. Contrairement à l'asservissement visuel 2D qui utilise les cordonnées de la projection de la cible dans le plan de l'image, l'asservissement 3D se base sur une mesure 3D de la cible dans la scène.

Historiquement, l'obtention des mesures 3D nécessitait une reconstruction 3D de l'information visuelle. Cette dernière est basée sur une estimation de la position relative entre la caméra et l'objet dans la scène. Cette estimation est déduite à partir des données visuelles extraites de l'image observée par conséquent, la connaissance d'un modèle tridimensionnel de la cible est nécessaire.

De ce fait, la reconstruction 3D engendre une complexité de calcul s'accroissant avec la complexité de la cible. Cependant le développement des capteurs extéroceptifs (laser, ultrason,…) et surtout la camera RGB_D appelée Kinect a permis de résoudre ce problème aisément. En effet, ce dispositif fournit en ligne la mesure de la profondeur grâce aux algorithmes de perception embarqués sur ce dispositif.

En utilisant les informations 3D, l'asservissement 3D repose sur plusieurs techniques qui se différencient par le type des signaux à réguler et la manière de les exploiter.

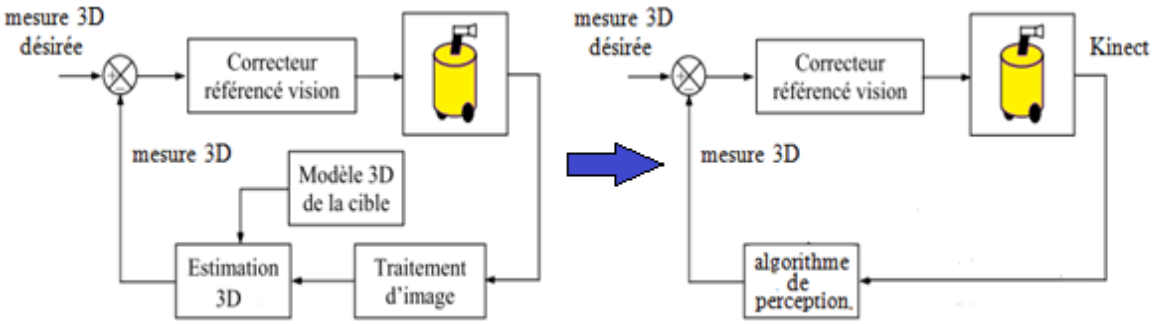


Fig.1.4 Boucle d'asservissement 3D

Parmi ces techniques, on distingue celles utilisant les primitives géométriques extraites de la cible (point, segment de droit, portion d'ellipse…) dans la plupart des cas il s'agit de points. Tout d'abord la fonction de la tâche est définie en fonction des coordonnées 3D de ces points puis la régulation a pour objet d'assurer la convergence à zéro de la fonction tache donc les points courants sont amenés vers les points désirés. Cette technique est souvent appelé « asservissement visuel 3D point ».

D'autres méthodes privilégient la régulation de la pose de la cible dans la scène par rapport à la Kinect (position et orientation). Pour ces méthodes l'objectif de commande consiste à faire converger le repère lié à la cible vers un repère désiré ou encore le repère de la Kinect vers un repère désiré. Cette convergence peut être réalisée de différentes manières par exemple par un retour d'état [Martinet, 1999] qui sera détaillé au chapitre 2.

L'inconvénient majeur de l'asservissement visuel 3D est que la cible peut sortir du champ de vision de la Kinect pendant la régulation car l'asservissement ne s'effectue pas au niveau de l'image.

3.3 Asservissement visuel 2D$\frac{1}{2}$

L'asservissement visuel 2D$\frac{1}{2}$ a été introduit par ***Chaumette et al*** [Chaumette, 2002]et il est considéré comme une variante de l'asservissement 2D. En effet, il utilise une combinaison d'informations ou certaines d'entre elles sont dans l'image et d'autres dans l'espace capteur.

Cette technique est particulièrement intéressante car elle ne présente pas les inconvénients du 2D ni ceux du 3D.

3.4 Asservissement visuel d2D/dt

Ce type d'asservissement visuel se base sur la régulation de la vitesse relative entre la cible et la caméra. La référence est définie comme un champ de vitesse désiré des indices visuels dans le plan image. Le principe de la commande consiste alors à contrôler les mouvements de la caméra de telle sorte que le mouvement 2D mesuré atteigne un champ de vitesse désiré d'où l'appellation d'asservissement visuel d2D/dt. Contrairement aux techniques présentées précédemment qui contraignent les vitesses du robot pour pouvoir asservir les positions liées à la cible, cette technique contraint, en plus des vitesses, les accélérations afin d'asservir la vitesse des primitives.

4. Conclusion

Ce premier chapitre a été entamé par la présentation de l'asservissement visuel en tant que commande référencée vision avec une brève explication de la commande référencée capteur ainsi que le formalisme de la fonction de tache. Par la suite, nous avons cité les différentes techniques de l'asservissement visuel et détaillé les plus répandues à savoir l'asservissement visuel 2D et l'asservissement visuel 3D dont la différence majeur est la nature de l'information visuelle en amont de la boucle de réglage.

Le prochain chapitre sera consacré à l'application de ces deux dernières techniques sur notre robot mobile afin de satisfaire une tâche de poursuite d'une cible.

CHAPITRE 2

Modélisation du système robotique et commandes

1. Introduction

D'un point de vue méthodologique, l'asservissement visuel consiste à intégrer dans la boucle de commande des robots, des informations visuelles fournies par des caméras afin de réaliser l'action souhaitée permettant à un robot de réaliser des tâches de diverses natures.
Dans notre cas, nous proposons d'étudier le problème de l'utilisation de ces techniques pour implémenter une tâche d'asservissent visuel sur un robot mobile afin de contrôler le mouvement d'un système dynamique dans notre cas une personne en mouvement en utilisant un capteur RGB-D.

Dans ce chapitre, nous présenterons la plateforme expérimentale, le capteur de vision ainsi que les notions de géométrie projective qui sont un outil mathématique bien adapté pour décrire les relations entre la scène tridimensionnelle observée et son image.
Nous détaillerons par la suite la commande utilisée ainsi que les outils nécessaires pour élaborer cette dernière. La commande est synthétisée dans un premier temps en considérant la tache comme un problème de régulation dans l'image définie dans l'espace 2D puis en second lieu la question est vue comme une tache de positionnement de repères tridimensionnels. En se basant sur la mesure 3D de la cible dans la scène, la commande 3D est mise en œuvre de deux manières.

2. Présentation du Robot mobile B21r

Commençons par présenter l'élément clé de notre travail soit la plateforme robotique : le robot mobile B21R est une plateforme expérimentale fabriquée par la société américaine IRobot et mise à notre disposition par le CDTA.
Ce robot est destiné à se déplacer dans un milieu d'intérieur, il a une forme cylindrique pouvant se déplacer grâce à deux moteurs indépendants. Celui destiné à la rotation peut changer l'orientation des quatre roues du robot avec une vitesse entre 1deg/sec et 90deg/sec et le moteur de translation peut faire tourner les roues avec une vitesse variant entre 1cm/s et 90cm/s. Le B21r est dit non holonome car il dispose de 2 degrés de liberté sur un plan et puisque les translations latérales sont impossibles à réaliser et sachant que les vitesses peuvent être négatives les mouvements possibles sont : la translation (avance ou recule) et la rotation (tourne vers la droite ou vers la gauche).

Fig.2.1 : Robot mobile B21r

3. Modélisation du système robotique

La détermination de la commande d'un système donné, nécessite le plus souvent la connaissance d'un modèle du système. Cette modélisation doit être menée avec rigueur, en vue d'obtenir le modèle le plus proche de la réalité afin de garantir par la suite de meilleures performances en termes de stabilité et de précision.

En effet, nous allons présenter une modélisation explicite et détaillée de la plateforme robotique utilisée. Pour ce faire nous définissons tout d'abord les repères suivants :

- le repère lié à la scène $R_O(O,X_O,Y_O, Z_O)$;
- le repère lié à la base mobile $R_M(M,X_M, Y_M, Z_M)$;
- et le repère lié à la Kinect $R_k(k,X_k, Y_k , Z_k)$.

De plus, nous notons par (t_x, t_y) les coordonnées de l'origine du repère lié à la Kinect dans le repère mobile.

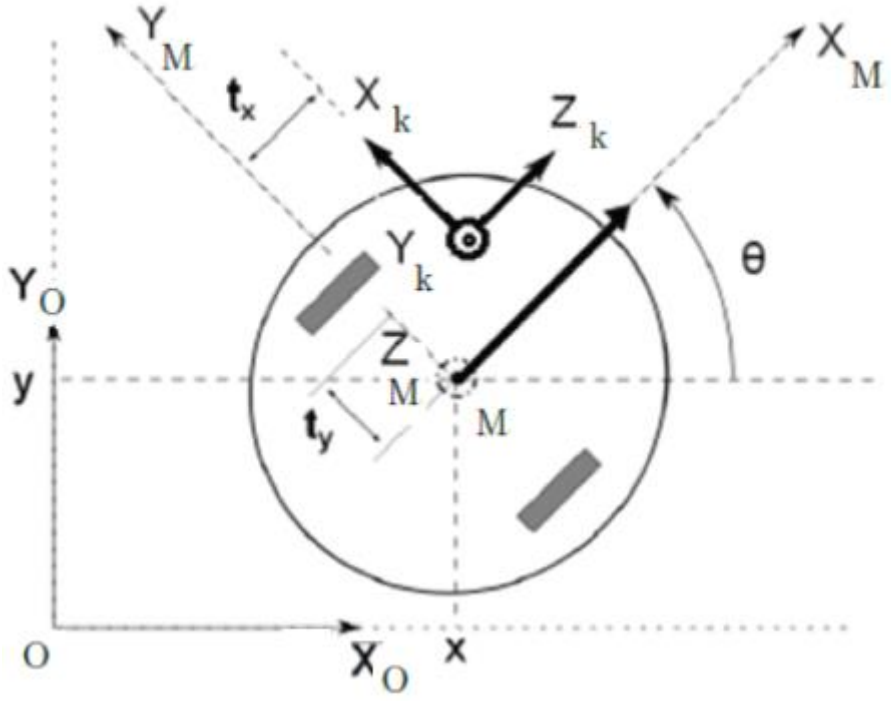


Fig.2.2 : Repères du système Robot-Kinect

La position du robot mobile est définie par les coordonnées (x, y) du point M dans le repèreR_O, par contre son orientation est repérée par l'angle ϑ entre l'axe X_O et l'axeX_M. En outre la position relative de la camera dans la base mobile est définie par le vecteur :

$$\overrightarrow{MK} = \begin{pmatrix} t_x \\ t_y \\ 0 \end{pmatrix} \qquad (2.1)$$

Le vecteur configuration du robot est défini comme suit :

$$q = \begin{pmatrix} l \\ \vartheta \end{pmatrix} \qquad (2.2)$$

Où l représente l'abscisse curviligne du robot dans le repère de la scène ainsi la dérivée par rapport au temps du vecteur configuration représente le vecteur de commande.

Dans la mesure où notre système robotique est commandable en vitesse, il y a lieu de définir la relation entre le torseur cinématique du robot noté T_{K/R_o} et le vecteur commande notéq̇.Par définition, ce torseur cinématique est constitué des composantes des vitesses de translation et de rotation ; il est alors comme suit :

$$\mathbf{T_{K/R_o}} = (\mathbf{V_{K/R_o}}, \Omega_{\mathbf{K/R_o}})^{\boldsymbol{T}} \qquad (2.3)$$

Avec :

$$\mathbf{V}_{\mathbf{K}/R_o} = (\,\mathbf{V}X_K\ \ \mathbf{V}Y_K\ \ \mathbf{V}Z_K\,)^T \tag{2.4}$$

$$\Omega_{K/R_o} = (\Omega\, X_K\ \ \Omega\, Y_K\ \ \Omega Z_K\,)^T \tag{2.5}$$

La vitesse$\mathbf{V}_{k/Ro}^{R_K}$ est déduite en exploitant la loi de composition des vitesses :

$$V_{K/Ro}^{Rk} = V_{K/RM}^{Rk} + V_{M/Ro}^{Rk} + \Omega_M^{Rk} \wedge \overrightarrow{RC}^{Rk} \tag{2.6}$$

La Kinect est repérée par le vecteur $\overrightarrow{RC}^{Rk} = (t_y \quad 0 \quad t_x)^T$et commeelle est fixée sur le robot par conséquent sa vitesse, par rapport au repère lié au robot, est nulle et donc :

$$V_{k/RM}^{Rk} = \frac{d}{dt}\,(\overrightarrow{RC})^{Rk} = 0 \tag{2.7}$$

En outre, le robot est repéré dans la scène par le vecteur $\overrightarrow{OM}$ et sa vitesse par rapport à la scène est alors donnée par :

$$V_{M/Ro}^{Rk} = \frac{d\overrightarrow{OM}^{Rk}}{dt} \tag{2.8}$$

Donc :

$$V_{M/Ro}^{Rk} = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix} \tag{2.9}$$

Sachant que $\overrightarrow{OM}^{Rk} = (0 \quad 0 \quad l)^T$ et $v = \frac{dl}{dt}$ le calcul du terme $\Omega_{M/Ro}^{RC}$ conduit à

$$\Omega_{M/Ro}^{RC} = \begin{pmatrix} 0 \\ \dot{\vartheta} \\ 0 \end{pmatrix} \tag{2.10}$$

En remplaçant les termes dans l'équation on obtient le vecteur suivant :

$$V_{K/Ro}^{Rk} = \begin{pmatrix} \dot{\vartheta} tx \\ 0 \\ v - \dot{\vartheta} ty \end{pmatrix} \tag{2.11}$$

Par conséquent, le torseur cinématique de la Kinect, par rapport à Ro et projeté dans le repère Kinect, est exprimé par:

$$\begin{pmatrix} VX_K \\ VY_K \\ VZ_K \\ \Omega X_K \\ \Omega Y_K \\ \Omega Z_K \end{pmatrix} = \begin{pmatrix} 0 & tx \\ 0 & 0 \\ 1 & -ty \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v \\ \dot{\vartheta} \end{pmatrix} \quad (2.12)$$

Donc la matrice Jacobienne est telle que :

$$J = \begin{pmatrix} 0 & tx \\ 0 & 0 \\ 1 & -ty \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \quad (2.13)$$

La forme réduite du Jacobien est obtenue en ne considérant que les mouvements réalisables par la Kinect donc ne sont prises en compte que la translation le long de X_k et celle le long de l'axe Z_k en plus de la rotation autour de l'axe Y_k d'où on obtient :

$$J_r = \begin{pmatrix} 0 & tx \\ 1 & -ty \\ 0 & 1 \end{pmatrix} \quad (2.14)$$

4. Système de Vision

4.1 Description de la Kinect

Le capteur Kinect est une barre horizontale reliée à une petite base avec un pivot motorisé, conçu pour être placé au-dessus ou en dessous de l'affichage vidéo. Le dispositif comporte une caméra RGB, un capteur de profondeur constitué par un projecteur laser et une caméra infrarouge (IR) et un micro phone comme le montre la **Figure 2.3.**

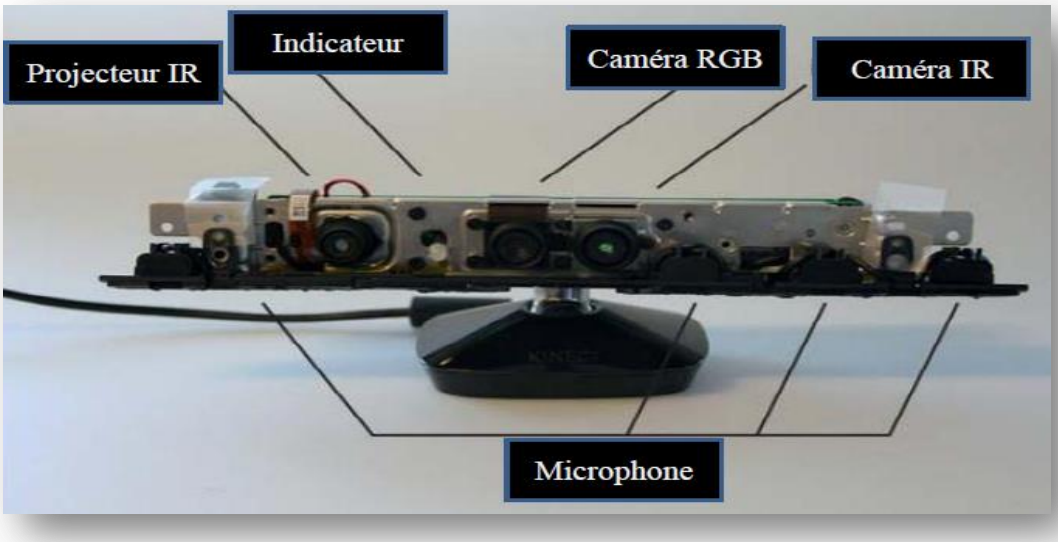


Fig.2.3: La Kinect sans cache

Le capteur de profondeur se compose d'un projecteur laser infrarouge combiné à une caméra IR. C'est un capteur du type CMOS (Complementary Metal Oxide Semiconductor) monochrome, qui capture des données vidéo en 3D.

4.2 Principe de fonctionnement de la Kinect

Le principe de fonctionnement de la Kinect est comme suit : un flux laser, dans la gamme infrarouge, est projeté sur la scène et simultanément un capteur CMOS monochrome, équipé d'un filtre IR (infrarouge), capture une image IR. En décodant la déformation des flux émis, le processeur de la Kinect exploite la position relative des points dans le modèle de la Kinect pour calculer la profondeur, qui correspond à chaque pixel dans l'image IR.

.

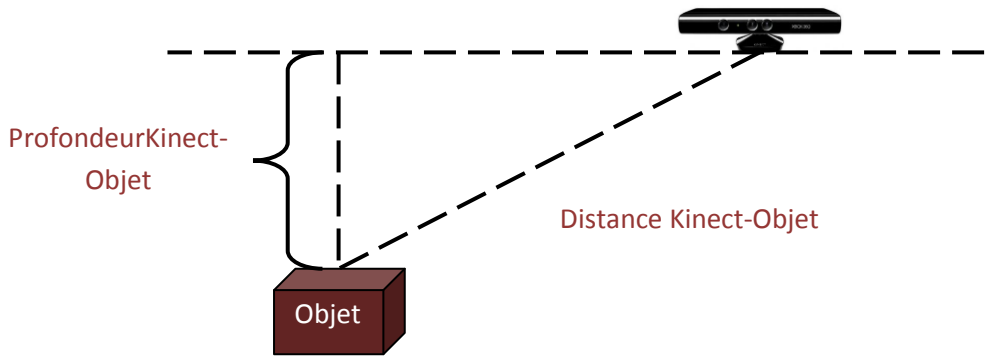


Fig.2.4 : la technique de mesure de la profondeur

Ainsi les valeurs de profondeur représentent la distance entre l'objet et le plan de la caméra IR (Fig. 2.4). De plus, le capteur de profondeur est considéré comme un dispositif qui peut fournir les coordonnées cartésiennes (*x, y, z*) des objets 3D, qui par la suite en calibrant les deux images IR et RGB, récupère les couleurs des objets en question.

4.3 Récupération de l'information 3D

Comme le capteur dispose de deux caméras dites IR et RGBpar conséquent, nous associons à chacune d'elle un repère et donc, un point de l'objet cible sera projeté dans ces deux repères par projection perspective.

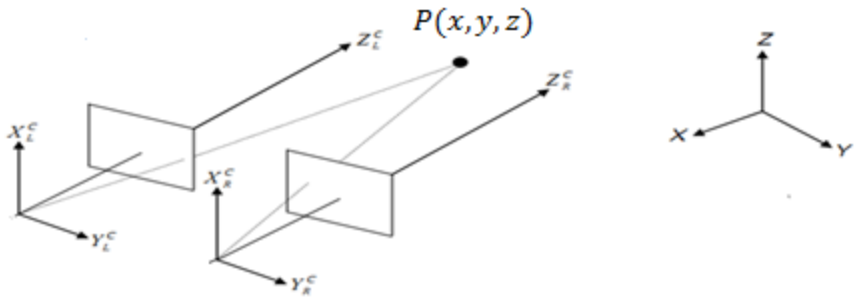


Fig.2.5 : Les deux repères de la Kinect

4.3.1 Projection perspective

Le modèle le plus couramment utilisé en vision par ordinateur est le modèle sténopé de la caméra appelé aussi trou d'épingle, comme il est détaillé ci-après.

Dans ce modèle (Fig.2.6), le point I dans le plan image est appelé point principal. La droite perpendiculaire au plan image passant par I, constitue l'axe optique. Le point C placé sur l'axe optique à une distance f du plan image est le centre de projection, alors que f représente la distance focale de la caméra. Comme la focale utilisée est de faible dimension, le modèle sténopé admet comme hypothèse que tous les rayons passent par le centre optique C. De ce fait, les points sont projetés sur le plan image par une projection perspective. Ainsi, un point P dans le champ de vision de la caméra se projette dans l'image le long d'une droite passant par p et C.

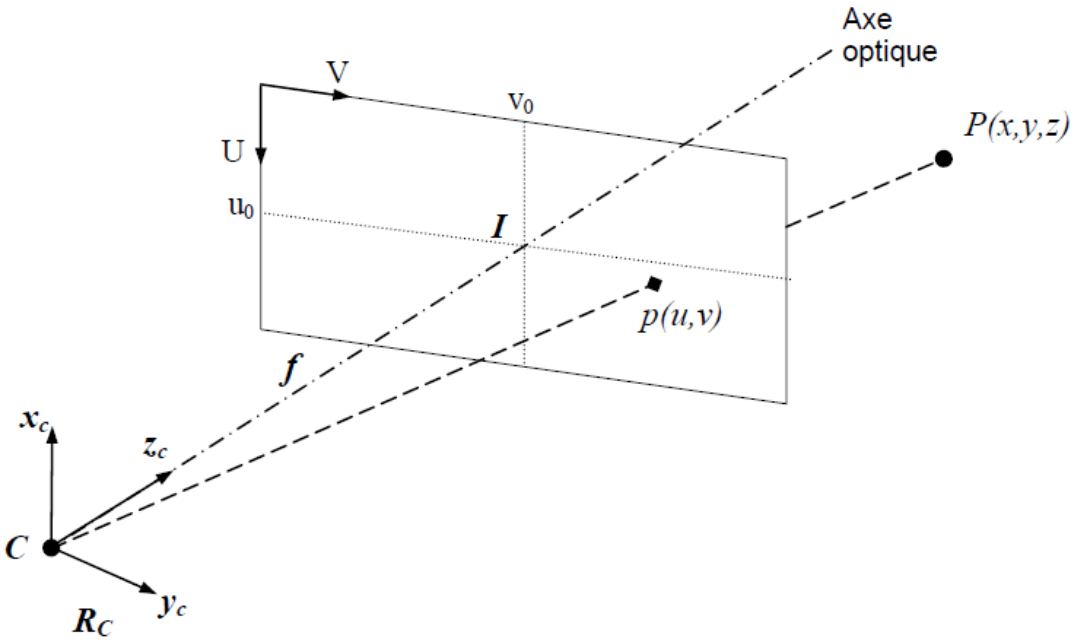


Fig.2.6 :Projection perspective

Dans le cas où le système de coordonnées lié à la caméra est représenté par le repère Rc (X_C, Y_C, Z_C) . En effet, l'origine de ce repère se trouve en C alors que le plan X_C-Y_C est parallèle au plan image, et l'axe $Z_C$est confondu avec l'axe optique. Soient (x, y et z) les coordonnées d'un point P dans le repère Rc ; dans ce même repère, les coordonnées (x', y' et z') de sa projection p sur le plan image sont exprimées comme suit :

$$\begin{cases} x' = f\frac{x}{z} \\ y' = f\frac{y}{z} \\ z' = f \end{cases} \qquad (2.15)$$

En outre ce même point p est mesuré en pixels par les coordonnés (u, v)dans un repère bidimensionnel U-V associé à l'image. La relation entre (u, v) et (x', y') fait intervenir les coordonnées en pixels de C dans le repère image (u_0, v_0), le facteur d'échelle horizontal k_u et celui vertical $k_v$sont tous deux exprimés en pixels/mm car les pixels d'une caméra sont rarement carrés. La transformation du repère caméra au repère image s'écrit (pour le point p) :

$$u = -k_u x' + u_0 \qquad (2.16)$$

$$v = +k_v y' + v_0 \qquad (2.17)$$

En remplaçant dans (2.16) et (2.17), (x′, y′) par leurs expressions (2.15), nous obtenons :

$$u = -k_u \, \mathrm{f}\frac{\mathrm{x}}{\mathrm{z}} + \mathrm{u}_0 \qquad (2.18)$$

$$v = k_v \, \mathrm{f}\frac{\mathrm{y}}{\mathrm{z}} + \mathrm{v}_0 \qquad (2.19)$$

En posant $\mathrm{d} = \mathrm{z}$ on obtient

$$\mathrm{d}u = -k_u \mathrm{fx} + \mathrm{du}_0 \qquad (2.20)$$

$$\mathrm{d}v = k_v \mathrm{fy} + \mathrm{dv}_0 \qquad (2.21)$$

D'où l'écriture matricielle :

$$\begin{pmatrix} du \\ dv \\ d \end{pmatrix} = \begin{pmatrix} -k_u \mathrm{f} & 0 & \mathrm{u}_0 \\ 0 & k_v \mathrm{f} & \mathrm{v}_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad (2.22)$$

En posant $\alpha_u = -k_u \mathrm{f}$ et $\alpha_v = k_v \mathrm{f}$ le système prend la forme :

$$\begin{pmatrix} du \\ dv \\ d \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & \mathrm{u}_0 \\ 0 & \alpha_v & \mathrm{v}_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad (2.23)$$

$$\begin{pmatrix} du \\ dv \\ d \end{pmatrix} = Ic \begin{pmatrix} x \\ y \\ z \end{pmatrix} \text{et} Ic = \begin{pmatrix} \alpha_u & 0 & \mathrm{u}_0 \\ 0 & \alpha_v & \mathrm{v}_0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (2.24)$$

Donc pour une caméra donnée (c), la transformation perspective entre l'espace projectif et le plan projectif et une application linéaire exprimée par la matrice Ic qui est fonction des paramètres intrinsèques qui sont les facteurs d'échelles de chaque caméra et les paramètres extrinsèques qui sont la matrice de rotation translation et de la caméra IR par rapport à la caméra RGB.

4.3.2 Reconstruction 3D

La reconstruction du nuage 3D nécessite plusieurs étapes. Nous admettons que les deux caméras (IR et RGB) sont préalablement calibrées donc nous disposons des paramètres $(\alpha_u, \alpha_v, \mathrm{u}_0, \mathrm{v}_0)$ de ces caméras. Les étapes à suivre sont comme suit :

L'acquisition de l'image de profondeur (Depth) à travers la caméra IR et la connaissance de son calibrage permettent de construire le nuage3D (N_{IR}) dans le repère de la caméra IR (projection perspective) :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = {I_{IR}}^{-1} \begin{pmatrix} du \\ dv \\ d \end{pmatrix} \text{ ou encore } N_{IR} = {I_{IR}}^{-1} \begin{pmatrix} du \\ dv \\ d \end{pmatrix} \tag{2.25}$$

L'inverse de la matrice I_{IR} est donnée par :

$${I_{IR}}^{-1} = \begin{pmatrix} \frac{1}{\alpha_u} & 0 & -\frac{1}{\mathrm{u}_0} \\ 0 & \frac{1}{\alpha_v} & -\frac{1}{\mathrm{v}_0} \\ 0 & 0 & 1 \end{pmatrix} \tag{2.26}$$

La connaissance de l'information sur la couleur de chaque point nécessite la reconstruction du nuage 3D (N_{RGB}) fournit par la caméra RGB.

Pour la reconstruction du nuage 3D (N_{RGB}) dans le repère de la caméra RGB, nous utilisons les résultats du calibrage stéréo entre les deux caméras IR et RGB. La transformation géométrique entre les points est donnée par :

$$N_{RGB} = R_{EX} * N_{IR} + T_{EX} \tag{2.27}$$

$$N_{RGB} = \begin{bmatrix} X1 \\ Y1 \\ Z1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.28}$$

Ou T_{EX} est le vecteur de translation et R_{EX} est la matrice de rotation.

4.4 Reconnaissance physique

L'exécution de la tache de suivi d'une cible par asservissement visuel passe par l'étape de la reconnaissance des points décrivant cette dernière parmi le nuage 3D observé par la Kinect. Il est à noter que la Kinect détecte un squelette (ensemble de points particuliers liés à une personne) en guise de personne. La Kinect, mise à notre disposition, peut détecter au maximum jusqu'à six squelettes des personnes assises ou debout. Cette caractéristique représente l'une des principales limites actuelles du périphérique. Parmi ces six squelettes, deux seulement peuvent être reconnus totalement : on parle alors de squelettes « trackés » et les quatre autres sont dits squelettes « non-trackés ». L'image ci-dessus montre cette fonctionnalité.

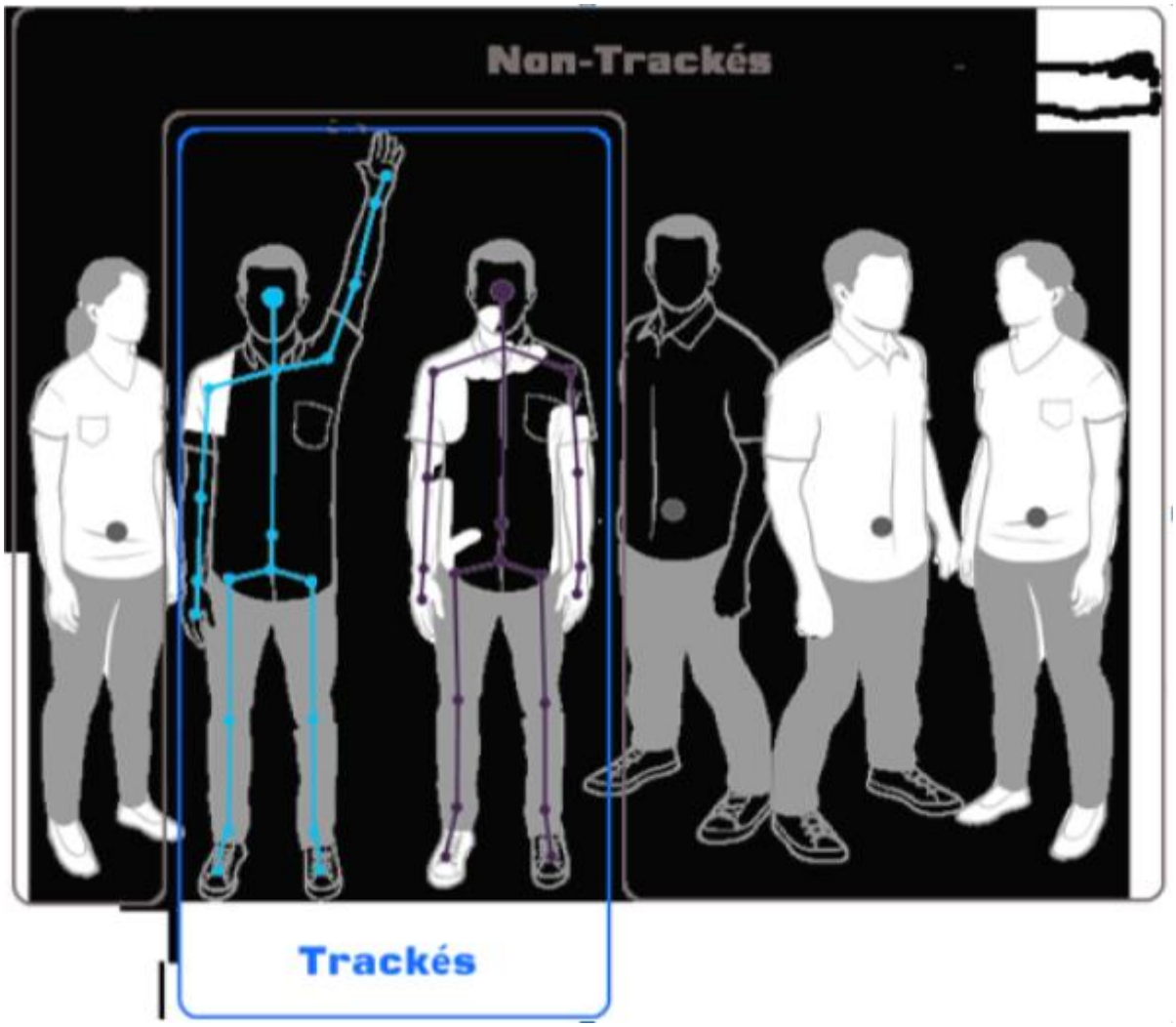


Fig.2.7 : Squelettes « trackés » et « non-trackés »

De la Figure 2-7, nous constatons que les squelettes « non-trackés »sont gérés différemment par rapport à ceux « trackés ». Dans le premier cas, la Kinect reconnait un seul point

correspondant à la position courante du squelette dans son repère alors que dans le second cas, elle détecte 20 points représentant le squelette de la personne détectée : c'est pourquoi nous parlons de squelettes et non de personnes.

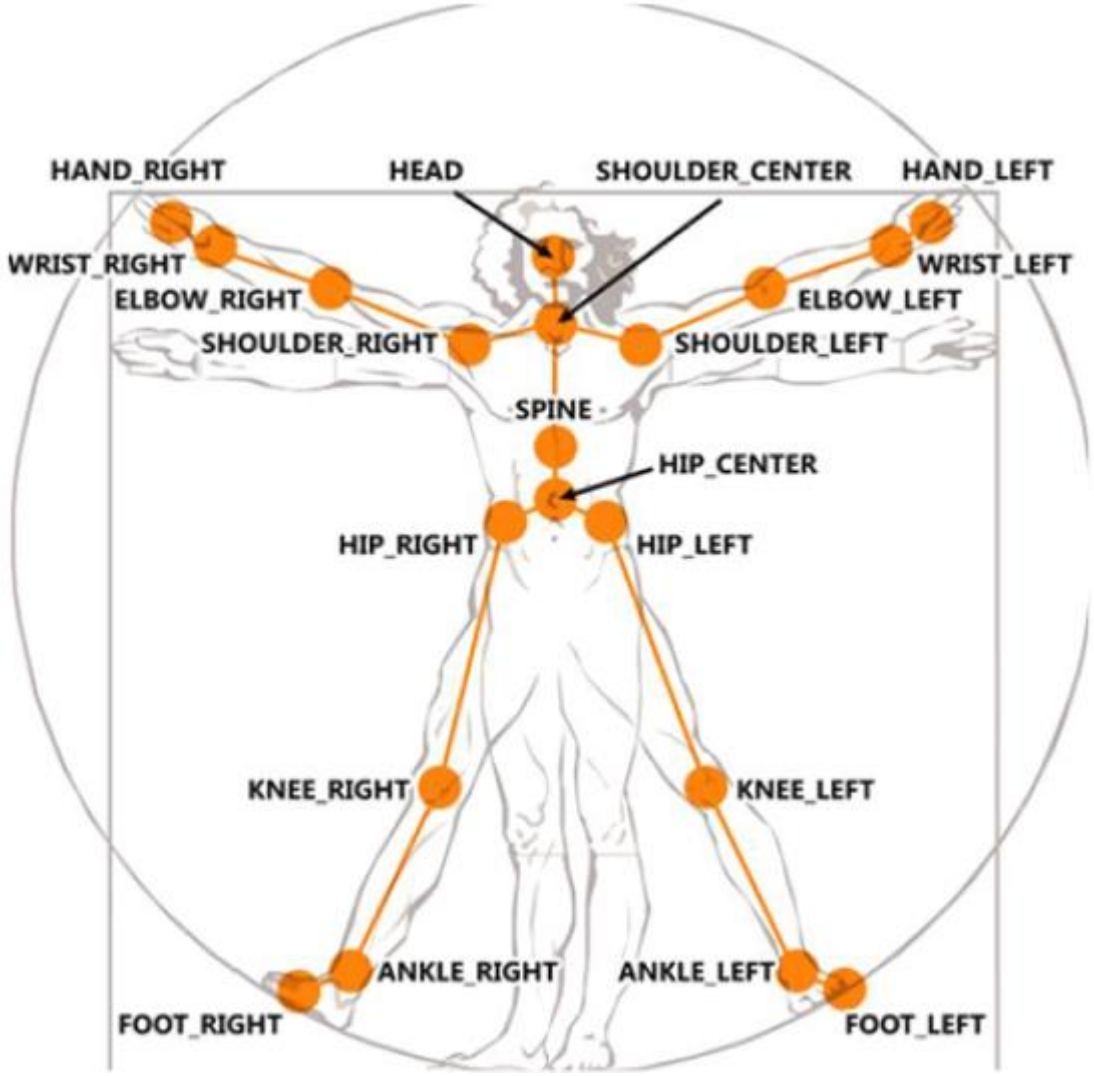


Fig.2.8 : Représentation d'un squelette tracké

Ces points sont appelés « joints ». Ils correspondent pour la plupart d'entre eux aux articulations de l'Homme et constituent le squelette de la personne.

La Figure 2.9 détaille les 20 joints d'un squelette tracké. Dans le cas d'un squelette non-tracké, un seul joint est pris en compte : HIP_CENTER. Chaque joint, associé à un squelette, est identifié par sa position en X, Y et Z dans le repère 3D ainsi que son orientation par les quaternions dans ce même repère 3D.

5. Asservissement visuel 2D

Une fois les points désirés sont récupérés, notre objectif est d'appliquer la commande référencée vision à notre robot mobile de manière à atteindre un but défini à priori dans l'image. Rappelons, que le principe de l'asservissement visuel 2D consiste à contrôler le mouvement de la caméra de manière à faire converger les indices visuels courants s(q,t) vers les indices visuels désirés s* . Pour cela, nous nous basons sur le formalisme des fonctions de tâche où couramment on utilise la fonction de tâche suivante :

$$e = C(s - s^*) \tag{2.29}$$

Où C représente une matrice de combinaison permettant de prendre en compte un nombre d'indices visuels supérieurs au nombre de degré de liberté du robot. Dans notre cas, le nombre de degrés de liberté est limité à 3, par conséquent C est pris de dimension 4 relative à 4 jonctions : la tête, le torse, l'épaule gauche et l'épaule droite.

Afin d'assurer la convergence de la fonction de tâche vers zéro, nous lui imposons une décroissance exponentielle. Par conséquent, la dynamique de la boucle fermée désirée est décrite par :

$$\dot{e} = -\lambda e \tag{2.30}$$

Où la matrice diagonale des gains positifs est donnée par :

$$\lambda = \begin{pmatrix} \lambda_{11} & 0 & 0 \\ 0 & \lambda_{22} & 0 \\ 0 & 0 & \lambda_{33} \end{pmatrix} \tag{2.31}$$

Où les termes diagonaux permettent la maitrise de la vitesse de convergence.
En outre :

$$\dot{e} = C\dot{s}(q,t) \tag{2.32}$$

Les informations visuelles s(q,t) caractérisent les mesures fournies par la Kinect. Celles-ci varient en fonction de la configuration du robot q c'est-à-dire implicitement en

fonction de ***r*** la position relative du robot par rapport à la scène et en fonction du temps *t* pris en tant que paramètre indépendant.

En différentiant s(q,t) par rapport au temps tout en incluant la variable q pour faire apparaitre le vecteur commande, il est possible d'exprimer la variation des informations visuelles en fonction des mouvements de la caméra :

$$\dot{s}(q,t) = \frac{\partial s}{\partial q}\frac{dq}{dt} + \frac{\partial s}{\partial t} = \frac{\partial s}{\partial r}\frac{\partial r}{\partial q}\frac{\partial q}{\partial t} + \frac{\partial s}{\partial t} \qquad (2.33)$$

Où :

- $\frac{\partial s}{\partial r}$ est la matrice d'interaction ou jacobéenne de l'image, notée L(s,z), déjà définie au chapitre 1.
- $\frac{\partial r}{\partial q}$ est la jacobéenne du robot ne dépendant que de la géométrie du robot et de de sa configuration. Elle a été ici définie entièrement au paragraphe 3 de ce chapitre 2.
- $\frac{\partial s}{\partial t}$ est un terme dû au mouvement propre de l'objet par rapport à la Kinect.

Dans la mesure où le pas de calcule de notre commande est inférieur à 10^{-4} seconde on peut considérer que la cible est immobile car sa vitesse est très petite par rapport à la vitesse de calcul de la commande ce qui donne $\frac{\partial s}{\partial t}$ =0.

- **Calcule de la Matrice d'interaction**

La matrice **L(s,z)** représente le lien entre le mouvement de la caméra et l'évolution des indices visuels. Elle dépend non seulement de la nature des informations visuelles choisies, mais aussi de la situation de la caméra par rapport à l'objet observé, ou, plus précisément de la distance entre la caméra et la cible observée. L'expression analytique de la matrice d'interaction peut être formulée en fonction du modèle géométrique de la cible et des primitives utilisées.

Ainsi, F. Chaumette a proposé une méthode générale de calcul analytique de la matrice pour différents types de primitives qui dans notre sont des points.

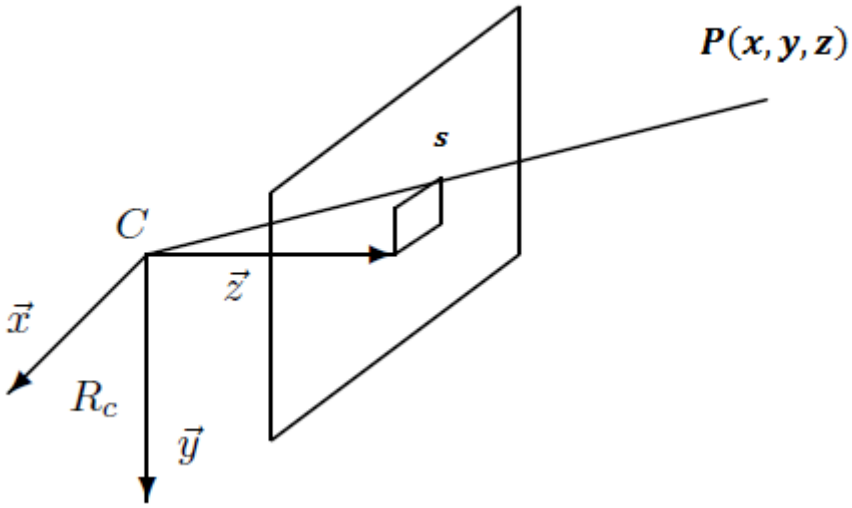


Fig.2.9 : Modèle de la camera

En effet, soit un point P de coordonnées $p(x, y, z)$ dans R_k (Fig.2.9), dont la projection perspective sur le plan image est un point s de coordonnées métriques (X, Y) et $\mathrm{T}_{\mathrm{K}/R_{\mathrm{o}}} = (\mathrm{V}_{\mathrm{K}/R_{\mathrm{o}}}, \Omega_{\mathrm{K}/R_{\mathrm{o}}})^{T}$ le torseur cinématique du robot ; la matrice d'interaction ***L(s,z)*** se déduit des équations du flot optique comme suit :

Indépendamment de la configuration, la vitesse de la cible P (exprimée dans le repère de la Kinect) s'obtient, en fonction du torseur cinématique, en exploitant la loi de composition des vitesses suivante :

$$\dot{P} = -\mathrm{V}_{\mathrm{K}/R_{\mathrm{o}}} - \Omega_{\mathrm{K}/R_{\mathrm{o}}} \wedge P \qquad (2.34)$$

En utilisant $\mathrm{V}_{\mathrm{K}/R_{\mathrm{o}}} = (\,\mathrm{V}X_{\mathrm{K}}\ \mathrm{V}Y_{\mathrm{K}}\ \mathrm{V}Z_{\mathrm{K}}\,)^{T}$ et $\Omega_{\mathrm{K}/R_{\mathrm{o}}} = (\Omega\, X_K\ \ \Omega\, Y_K\ \ \Omega Z_K\,)^{T}$, il vient :

$$\dot{x} = -\mathrm{V}X_{\mathrm{K}} - \Omega\, Y_K z + \Omega\, Z_K y \qquad (2.35)$$

$$\dot{y} = -\mathrm{V}Y_{\mathrm{K}} + \Omega\, X_K z - \Omega\, Z_K x \qquad (2.36)$$

$$\dot{z} = -\mathrm{V}Z_{\mathrm{K}} - \Omega\, X_K y + \Omega\, Y_K x \qquad (2.37)$$

La relation entre les coordonnés réelles et métriques est donnée par :

$$\begin{cases} X = \frac{x}{z} \\ Y = \frac{y}{z} \end{cases} \tag{2.38}$$

La dérivée temporelle de la relation précédente conduit à :

$$\begin{cases} \dot{X} = \frac{\dot{x}z - x\dot{z}}{z^2} = \frac{\dot{x}}{z} - x\frac{\dot{z}}{z^2} \\ \dot{Y} = \frac{\dot{y}z - y\dot{z}}{z^2} = \frac{\dot{y}}{z} - y\frac{\dot{z}}{z^2} \end{cases} \tag{2.39}$$

En remplaçant par le résultat des compositions de vitesse on obtient :

$$\begin{cases} \dot{X} = -\frac{\mathrm{V}X_{\mathrm{K}}}{z} + \frac{x\mathrm{V}Z_{\mathrm{K}}}{z^2} + \frac{xy}{z^2}\Omega\, X_K - \left(1 + \frac{x^2}{z^2}\right)\Omega\, Y_K + \frac{y}{z}\Omega\, Z_K \\ \dot{Y} = -\frac{\mathrm{V}Y_{\mathrm{K}}}{z} + \frac{y\mathrm{V}Z_{\mathrm{K}}}{z^2} - \frac{xy}{z^2}\Omega\, Y_K + \left(1 + \frac{y^2}{z^2}\right)\Omega\, X_K - \frac{x}{z}\Omega\, Z_K \end{cases} \tag{2.40}$$

Ou encore :

$$\begin{cases} \dot{X} = -\frac{\mathrm{V}X_{\mathrm{K}}}{z} + \frac{X\mathrm{V}Z_{\mathrm{K}}}{z} + XY\Omega\, X_K - (1 + X^2)\Omega\, Y_K + Y\Omega\, Z_K \\ \dot{Y} = -\frac{\mathrm{V}Y_{\mathrm{K}}}{z} + \frac{Y\mathrm{V}Z_{\mathrm{K}}}{z} - XY\Omega\, Y_K + (1 + Y^2)\Omega\, X_K - X\Omega\, Z_K \end{cases} \tag{2.41}$$

L'écriture de la relation précédente sous la forme matricielle conduit à

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} = \begin{pmatrix} -\frac{1}{z} & 0 & +\frac{X}{z} & XY & -(1+X^2) & Y \\ 0 & -\frac{1}{z} & +\frac{Y}{z} & +(1+Y^2) & -XY & X \end{pmatrix} \begin{pmatrix} \mathrm{V}X_{\mathrm{K}} \\ \mathrm{V}Y_{\mathrm{K}} \\ \mathrm{V}Z_{\mathrm{K}} \\ \Omega\, X_K \\ \Omega\, Y_K \\ \Omega\, Z_K \end{pmatrix} \tag{2.42}$$

or

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} = L(s,z)T_{K/R_o} \tag{2.43}$$

Donc

$$L(s,z) = \begin{pmatrix} -\frac{1}{z} & 0 & +\frac{X}{z} & XY & -(1+X^2) & Y \\ 0 & -\frac{1}{z} & +\frac{Y}{z} & +(1+Y^2) & -XY & X \end{pmatrix} \tag{2.44}$$

Ainsi, par le biais du torseur cinématique de la Kinect $\mathrm{T}_{\mathrm{K}/R_{\mathrm{o}}}$, nous pouvons relier la variation des informations sensorielles au mouvement réalisé par la Kinect, soit :

$$\dot{s}(q,t) = \mathrm{L}(s,z)\mathrm{T}_{\mathrm{K}/\mathrm{R}_{\mathrm{o}}} \tag{2.45}$$

On retrouve ici la notion d'action réflexe où les stimuli du capteur $\dot{s}$ sont aussitôt retranscrits par une action T_{K/R_o} .

Cette expression de la matrice d'interaction suppose que le torseur cinématique est de dimension (6×1), est entièrement défini par la relation (2.12)**.**

Or, nous sommes dans le cas où le torseur cinématique (T_{K/R_o}) est réduit et de dimension (3 ×1) ne contenant que les degrés de liberté réellement commandables. De ce fait, il est nécessaire de rendre cette matrice compatible en ne conservant que les colonnes correspondant à ces degrés de liberté. Nous obtenons alors la matrice d'interaction réduite $L_r(s,z)$suivante :

$$L_r(s,z)=\begin{pmatrix} L_r(X,z) \\ L_r(Y,z) \end{pmatrix}=\begin{pmatrix} -\frac{1}{Z} & 0 & Y \\ 0 & -\frac{1}{Z} & X \end{pmatrix} \quad (2.46)$$

Ce résultat est relatif à un seul point et il peut être étendu au cas où on dispose de quatre points. De ce fait, le vecteur des informations sensorielles est alors défini par :

$$s=(X_1,Y_1,X_2,Y_2,X_3,Y_3,X_4,Y_4)^T \quad (2.47)$$

où les couples (X_j,Y_j) représentent les coordonnées métriques du point $P_j$de la cible projetée dans le plan image de la caméra. La matrice d'interaction résultante est alors donnée par :

$$L_r(s,z)=(L_r(X1,z),L_r(Y1,z),L_r(X2,z),L_r(Y2,z),L_r(X3,z),L_r(Y3,z),L_r(X4,z),L_r(Y4,z))^T \quad (2.48)$$

Ayant la matrice d'interaction, il est possible d'exploiter $L_r(s,z)$ pour synthétiser une loi de commande réalisant le suivi du squelette d'une personne.

Rappelons qu'on a imposé une décroissance exponentielle à la fonction tache :

$$\dot{e}(q) = -\lambda e(q) \quad (2.49)$$

Ou encore

$$\begin{cases} \dot{e}(q) = -\lambda C(s-s^*) \\ \dot{e}(q) = CL_r(s,z)J_r\,\dot{q} \end{cases} \quad (2.50)$$

De la relation précédente, on en déduit :

$$CL_r(s,z)J_r\,\dot{q} = -\lambda C(s - s^*) \qquad (2.51)$$

La convergence de cette dernière relation exige que le terme $CL_r(s,z)$ soit défini positif car J_r est défini positif.

Une solution judicieuse, pour garantir la positivité de ce produit matriciel, consiste à prendre la matrice de combinaison Ccomme étant le pseudo-inverse à gauche de la matrice d'interaction, notée$L_r(s,z)^+$. Donc on peut écrire :

$$C = L_r(s,z)^+ = \left(L_r(s,z)^T L_r(s,z)\right)^{-1} L_r(s,z)^T \qquad (2.52)$$

Dans ce cas, la relation (2.50)se réduit à :

$$J_r\,\dot{q} = -\lambda L_r(s,z)^+(s - s^*) \qquad (2.53)$$

Soit :

$$\dot{q} = -{J_r}^+ \lambda L_r(s,z)^+(s - s^*) \qquad (2.54)$$

Ou ${J_r}^+$ est le pseudo inverse du jacobien J_r.

6. Asservissement 3D

Contrairement à la commande 2D qui agit au niveau de l'image, l'asservissement 3D considère la cible comme un modèle tridimensionnel situé dans le repère Kinect et donc impliquant sa position et son orientation.

Nous allons synthétiser une loi spécifique de commande dans le but d'amener la position de la cible relative à la Kinect sur une position désirée. Pour ce faire, nous proposons deux solutions. Pour la première solution, 4 points de la cible sont sélectionnés lesquels sont exprimés en 3D. Ainsi la mise en œuvre de la commande doit assurer la convergence des cordonnées courantes de ces 4 points vers les cordonnées désirées de ces mêmes points. Pour la deuxième solution, nous devons synthétiser une loi de commande qui amène le repère lié à la Kinect sur le repère désiré de la Kinect correspondant à la situation désirée de la cible.

Pour conclure, nous simulons le comportement du robot sous le contrôle des deux commandes.

6.1 Asservissement 3D point

Le formalisme du problème pour la première solution, nous conduit à imposer une fonction de tache, incorporant les informations 3D, de la forme suivante :

$$e(r,t) = C * (P - P^*) \tag{2.55}$$

Où

r est le vecteur configuration du robot,

C est une matrice prenant en compte un nombre de fonctionnalité en plus du degré de liberté du robot ;

P est un point de la cible, c'est la valeur courante de l'information fournit par les capteurs ;

P^*représente la pause de référence cible à atteindre dans le cadre du capteur.

Comme dans l'asservissement 2D la matrice d'interaction est définie par :

$$\mathcal{L}(p) = \frac{\partial P}{\partial r} = \frac{\partial P}{\partial t}\frac{\partial t}{\partial r} \tag{2.56}$$

Donc :

$$\frac{\partial P}{\partial t} = \mathcal{L}(p)\frac{\partial r}{\partial t} \tag{2.57}$$

Soient (***x, y, z)***les coordonnées cartésiennes dans le repère Kinect d'un point ***P*** de la cible, la vitesse du point P dans le repère de la Kinect est donnée par :

$$\dot{P} = -\mathrm{V}_{\mathrm{K}/R_{\mathrm{o}}} - \Omega_{\mathrm{K}/R_{\mathrm{o}}} \wedge P \tag{2.58}$$

Ou encore

$$\dot{x} = -\mathrm{V}X_{\mathrm{K}} - \Omega\, Y_K z + \Omega\, Z_K y \tag{2.59}$$

$$\dot{y} = -\mathrm{V}Y_{\mathrm{K}} + \Omega\, X_K z - \Omega\, Z_K x \tag{2.60}$$

$$\dot{z} = -\mathrm{V}Z_{\mathrm{K}} - \Omega\, X_K y + \Omega\, Y_K x \tag{2.61}$$

Ces relations peuvent s'écrire sous la forme matricielle suivante :

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{pmatrix}=\begin{pmatrix}-1 & 0 & 0 & 0 & -z & y\\ 0 & -1 & 0 & z & 0 & -x\\ 0 & 0 & -1 & -y & x & 0\end{pmatrix}\begin{pmatrix}\mathrm{V}X_{\mathrm{K}}\\ \mathrm{V}Y_{\mathrm{K}}\\ \mathrm{V}Z_{\mathrm{K}}\\ \Omega X_K\\ \Omega Y_K\\ \Omega Z_K\end{pmatrix} \quad (2.62)$$

Donc :

$$\mathcal{L}\,(\mathrm{p})=\begin{pmatrix}-1 & 0 & 0 & 0 & -z & y\\ 0 & -1 & 0 & z & 0 & -x\\ 0 & 0 & -1 & -y & x & 0\end{pmatrix} \quad (2.63)$$

Sachant que notre robot ne peut se déplacer que suivant X_K et Z_K et ne peut tourner qu'autour de Y_K par conséquent $\mathcal{L}\,(\mathrm{p})$ se réduit à la forme suivante :

$$\mathcal{L}\,(\mathrm{p})=\begin{pmatrix}\mathcal{L}x\\ \mathcal{L}y\\ \mathcal{L}z\end{pmatrix}=\begin{pmatrix}-1 & 0 & -z\\ 0 & 0 & 0\\ 0 & -1 & x\end{pmatrix} \quad (2.64)$$

Comme la ligne $\mathcal{L}y$ est nulle, on obtient au final :

$$\mathcal{L}_{\mathrm{r}}(\mathrm{p})=\begin{pmatrix}-1 & 0 & -z\\ 0 & -1 & x\end{pmatrix} \quad (2.65)$$

Dans notre exemple, la cible est détectée via quatre points relatifs aux deux épaules, à la tête et au torse par conséquent, le vecteur S est alors défini par :

$$\mathrm{S}=(P_1,P_2,P_3,P_4)^T=(X_1,Y_1,Z_1,X_2,Y_2,Z_2,X_3,Y_3,Z_3,X_4,Y_4,Z_4)^T \quad (2.66)$$

La matrice d'interaction globale est alors donnée par :

$$\mathcal{L}_{\mathrm{r}}(\mathrm{p})=\begin{pmatrix}\mathcal{L}_{\mathrm{r}}(P_1)\\ \mathcal{L}_{\mathrm{r}}(P_2)\\ \mathcal{L}_{\mathrm{r}}(P_3)\\ \mathcal{L}_{\mathrm{r}}(P_4)\end{pmatrix} \quad (2.67)$$

La relation entre la variation de P par rapport au temps et le torseur cinématique T_{K/R_o} de la Kinect est donnée par :

$$\dot{P} = \mathcal{L}_{\mathrm{r}}(\mathrm{p})\mathrm{T}_{\mathrm{K}/R_{\mathrm{o}}} \quad (2.68)$$

En combinant (2.54) et (2.67)et compte tenu de la convergence exponentielle pour le positionnement de l'ensemble des composantes de la fonction de tâche, nous obtenons l'expression de la commande suivante où$\mathcal{L}(\mathrm{p})^{+}$représente la matrice pseudo inverse de$\mathcal{L}(\mathrm{p})$:

$$\mathrm{T}_{\mathrm{K}/R_{\mathrm{o}}} = -\lambda \mathcal{L}_{\mathrm{r}}(\mathrm{p})^{+}(S - S^{*}) \quad (2.69)$$

$$\mathrm{T}_{\mathrm{K}/R_{\mathrm{o}}} = J_r\,\dot{q} \quad (2.70)$$

Où J_r est le Jacobien du robot

En combinant les deux dernières relations, on obtient :

$$J_r\,\dot{q} = -\lambda \mathcal{L}_{\mathrm{r}}(\mathrm{p})^{+}(S - S^{*}) \quad (2.71)$$

Donc

$$\dot{q} = -{J_r}^{+}\lambda \mathcal{L}_{\mathrm{r}}(\mathrm{p})^{+}(S - S^{*}) \quad (2.72)$$

6.2Asservissement 3D par retour d'état dans l'espace capteur

Cette fois-ci, la tache robotique à réaliser consiste à ramener le repère Kinect courant dans une situation désirée dite repère Kinect désirée. Cette tache peut être effectuée en deux étapes. La première étape consiste à définir le repère Kinect désirée à travers la pose désirée de la cible par rapport au robot (le mouvement de la cible étant libre on fait déplacer la Kinect pour atteindre cette pose). En deuxième étape, on calcule une commande qui fait converger le repère Kinect courant vers le repère Kinect de référence (Tamtsia, 2013).

Nous définirons tout d'abord 3 repères : Kinect désiré, Kinect courant, cible ainsi que les transformations homogènes associées:

- Mo(t) matrice de transformation homogène entre le repère Kinect désiré et la cible ;
- Mp(t) matrice de transformation homogène entre le repère Kinect courant et le repère de la cible ;

- Mc(t) matrice de transformation homogène entre le repère Kinect désiré et le repère Kinect courant.

De la Figure 2-10, on voit qu'il est possible de déduire la matrice Mc(t) de la manière suivante :

$$Mc(t) = MoMp(t)^{-1} \tag{2.73}$$

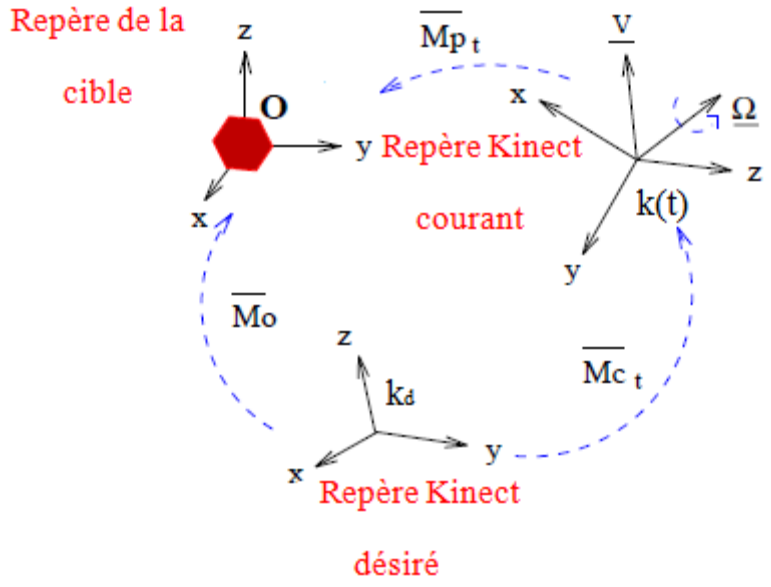


Fig.2.10 : Représentation des repères utilisés

En outre les paramètres de pose sont en général exprimés par une matrice de transformation homogène de la forme :

$$\overline{M}c(t) = \begin{pmatrix} R(t) & k_d k(t)_{k_d} \\ 0 & 1 \end{pmatrix} \tag{2.74}$$

Où R(t) représente la matrice rotation et $k_d k(t)_{k_d}$ la position du repère Kinect courant dans le repère Kinect désiré.

Pour le besoin des développements qui vont suivre, on exploite la représentation suivante de la matrice d'orientation (voir Annexe B):

$$R(t) = e^{(-AS(\vartheta))} \tag{2.75}$$

Ou $AS(\vartheta)$ est la matrice antisymétrique associée au vecteur d'orientation du repère Kinect courant par rapport au repère Kinect désiré.

$$\underline{\vartheta}(t) = \|\underline{\vartheta}\|.\underline{u}(t) = \begin{pmatrix}\vartheta_1\\ \vartheta_2\\ \vartheta_3\end{pmatrix} \tag{2.76}$$

Tout d'abord nous adoptons les notations suivantes :

$\underline{x(t)}$désigne la position courante de la Kinect dans le repère désiré ;

$\underline{V(t)} = \underline{V(t)}_{k_d}$représente le vecteur de translation de la Kinect exprimé dans le repère Kinect désiré;

AS($\underline{\Omega}$) =AS($\underline{\Omega}$)$_C$ est la matrice antisymétrique associée au vecteur de rotation $\underline{\Omega}$ exprimée dans l'espace de la Kinect.

Afin d'établir le modèle d'état du système on dérive(2.74), ce qui conduit à :

$$\begin{cases}\frac{dx}{dt} = R(t)V(t)\\ \frac{dR(t)}{dt} = R(t)As(\Omega)\end{cases} \tag{2.77}$$

Pour la représentation dans l'espace d'état nous choisissons le vecteur d'état suivant :

$$\underline{X}(t) = \left(\underline{x}^T(t), \underline{y}^T(t)\right)^T$$

où

$\underline{x}(t) = \begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}$ représente la position courante de la Kinect ;

$\underline{y}(t) = \begin{pmatrix}y_1\\ y_2\\ y_3\end{pmatrix}$ représente l'orientation courante de la Kinect

Pour calculer le vecteur de l'orientation on considère l'approximation $\alpha \simeq sin\alpha$ et dans le cas vectorielle on a ;

$$As\left(\underline{y}(t)\right) = \frac{1}{2} * \left(\mathrm{R}^{T}(t) - \mathrm{R(t)}\right) = \sin(\|\underline{\vartheta}\|).\, AS(\underline{u}(t)) \qquad (2.78)$$

Cette approximation est admissible pour $\|\underline{\vartheta}\| < \frac{\pi}{2}$.

Pratiquement, on calcule R(t) directement à partir des quaternions fournis par l'algorithme de reconnaissance physique en traitant les informations issues de la kinect.

$$R(t) = s^2 I_3 + 2\mathrm{s}(\overline{v}) + vv^T + (\overline{v})^2 \qquad (2.79)$$

Ce quaternion $Q = (\mathrm{w}, \mathrm{Qx}, \mathrm{Qy}, \mathrm{Qz})$est de norme unité

Où$\overline{v}$ est le tenseur antisymétrique associé au vecteur $v = (\mathrm{Qx}, \mathrm{Qy}, \mathrm{Qz})$ du quaternion. Dans cette relation Q doit être impérativement de norme unité.

Le vecteur $\underline{y}(t)$ peut être tiré de $As\left(\underline{y}(t)\right)$ via la relation suivante:

$$\underline{y}(t) = A_1 * As\left(\underline{y}(t)\right) * v_1 + A_2 * As\left(\underline{y}(t)\right) * v_2 + A_3 * As\left(\underline{y}(t)\right) * v_3 \qquad (2.80)$$

Avec :

$$As\left(\underline{y}(t)\right) = \begin{pmatrix} 0 & -y_3 & y_2 \\ y_3 & 0 & -y_1 \\ -y_2 & y_1 & 0 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} ; A_2 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} ; A_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$v_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} ; \; v_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} ; \; v_3 = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$$

Cette représentation, nous permet d'exprimer une équation d'état du système :

$$\frac{d}{dt}\underline{X}(t) = B\left(\underline{X}\right).\underline{U} \qquad (2.81)$$

Ou :

$$B(\underline{X}) = \begin{pmatrix} R(y) & O_3 \\ O_3 & A(y) \end{pmatrix} \quad (2.82)$$

$$\underline{U}(t) = \left(\underline{V}^T, \underline{\Omega}^T\right)^T \quad (2.83)$$

Cette expression de la matrice de commande suppose que le vecteur de commande soit de dimension (6×1). Or, dans notre cas, le vecteur commande est réduit, il est de dimension (3×1) ne tenant compte que des degrés de liberté réellement commandables. De ce fait, il est nécessaire de rendre cette matrice compatible avec notre système en ne conservant que les colonnes correspondant à ces degrés de liberté. Nous obtenons alors la matrice réduite $B(\underline{X})$ de dimension (6×3).

Pour assurer la convergence, on impose au système (2.81), la dynamique linéaire suivante :

$$\frac{d}{dt}\underline{X}(t) = -K\underline{X} \quad (2.84)$$

Tenant compte de (2.81) et (2.84), on en déduit que :

$$B(\underline{X}).\underline{U} = -K\underline{X} \quad (2.85)$$

Et donc on a :

$$\underline{U} = -B(\underline{X})^{+}\overline{K}\underline{X} \quad (2.86)$$

Où $B(\underline{X})^{+}$ représente la matrice pseudo inverse de $B(\underline{X})$ donnée par :

$$B(\underline{X})^{+} = (B(\underline{X})^T B(\underline{X}))^{-1} * B(\underline{X})^T \quad (2.87)$$

Et $\overline{K}$est une matrice de dimension (6×6).

Comme le vecteur de commande exprimé dans l'espace de la Kinect est donné par :

$$\underline{U} = \begin{pmatrix} VX_K \\ VZ_K \\ \Omega Y_K \end{pmatrix} \quad (2.88)$$

Par définition, le Jacobien du robot relie la vitesse de translation et de rotation du robot au vecteur de commande (représentant le tenseur cinématique de la Kinect) ainsi, on peut écrire :

$$\begin{pmatrix} VX_K \\ VZ_K \\ \Omega Y_K \end{pmatrix} = J_r \begin{pmatrix} v \\ \dot{\vartheta} \end{pmatrix} \quad (2.89)$$

Donc :

$$\begin{pmatrix} v \\ \dot{\vartheta} \end{pmatrix} = J_r{}^{+} \begin{pmatrix} VX_K \\ VZ_K \\ \Omega Y_K \end{pmatrix} \quad (2.90)$$

$$\dot{q} = J_r{}^{+} \underline{U} \quad (2.91)$$

7. Conclusion

Dans ce chapitre, nous avons présenté le principe de la commande référencée vision d'un robot mobile non-holonome muni d'une Kinect. Nous avons alors décrit brièvement le système robotique considéré ainsi que notre capteur de vision en rappelant les concepts fondamentaux qui sont les projections perspectives.

De plus, nous avons établi les modèles nécessaires pour exprimer le lien entre le mouvement de la caméra et la variation des indices visuels. Nous avons ensuite présenté le formalisme des fonctions de tâche puis, nous avons montré comment exploiter ce formalisme pour synthétiser deux lois de commande en boucle fermée. Une première loi a été établie en exploitant les informations visuelles extraites de l'image puis une deuxième loi assurant la convergence des points 3D de la cible vers des points désirés. La troisième étant une commande 3D où la cible est considéré comme un repère tridimensionnel situé par rapport au repère Kinect et donc impliquant sa position et son orientation. Ces trois lois de commandes sont compatibles avec la tache considérée dans notre étude à savoir : « le suivi d'une personne dans un milieu intérieur ».

CHAPITRE 3

Architecture du robot et Implémentation

1. Introduction

Au chapitre 2, nous avons exposé les aspects matériels liés à notre plateforme robotique (robot et Kinect) concernant sa modélisation et le principe de fonctionnement du système de vision, puis nous avons établi les lois de commande (2D et 3D) afin d'assurer la tâche d'asservissement visuel.

Dans ce chapitre, nous présentons l'architecture logicielle de notre système robotique, les différents programmes mis en jeu ainsi que les tests pour valider expérimentalement nos commandes

L'expérimentation des commandes proposées est prévue sur le robot B21r, mis à notre disposition au CDTA. Ce robot est d'ancienne génération où le PC embarqué, de très faibles performances, ne permet pas d'intégrer directement les capteurs externes. Par conséquent, il fallait une alternative pour connecter la Kinect et ainsi récupérer les données et par là assurer le calcul de la commande et l'exécution de la tâche robotique.

Pour cela, l'équipe Robotique du CDTA a tout simplement remplacé ce PC embarqué par une nouvelle architecture intégrant un PC. Ce dernier comporte d'une part, un outil robotique très consistant **ROS** et d'autre part, les programmes assurant la communication entre ce dernier et le système de commande du robot (dénommé **rFlex**) qui sera présenté dans la suite de ce chapitre.

2. L'architecture du B21r

L'exécution de notre tâche nécessite un PC externe ayant des caractéristiques lui assurant d'intégrer les informations issues de la **Kinect** et l'utilisation de l'outil robotique **ROS**. A ce PC externe a été connecté la **Kinect** via une connexion USB et le système **rFlex** via une liaison RS232.

Pour ce faire, il a fallu adapter l'architecture software afin d'intégrer les programmes assurant la communication avec le système **rFlex**. Ce dernier transmet, à son tour, les vitesses élaborées par les programmes de commande aux cartes de commande des moteurs afin que le robot puisse exécuter la tâche d'asservissement.

2.1 Architecture Hardware

L'architecture hardware de notre système robotique comporte essentiellement 4 dispositifs à savoir : les cartes de commandes des moteurs, la carte **rFlex**, le **PC** externe et la **Kinect**.

2.1.1 Le système rFlex

Le système **rFlex** est un système de contrôle conçu par la société **iROBOT**, il est constitué d'une carte de commande et d'un écran .La carte de commande **rFlex** est considérée comme l'élément central du robot car elle est connectée aux différents capteurs et moteurs. Cette carte est dotée de programmes ayant pour charge la récupération des données capteurs et la transmission des consignes aux moteurs. Quant à l'écran, il présente une interface offrant à l'utilisateur la possibilité de diriger le robot au moyen d'un menu facile à manipuler. Cet écran affiche également l'état du robot (batteries, luminosité,..).

Ce système est accessible via le **PC** externe car il est équipé d'un port de connexion série **RS232**.

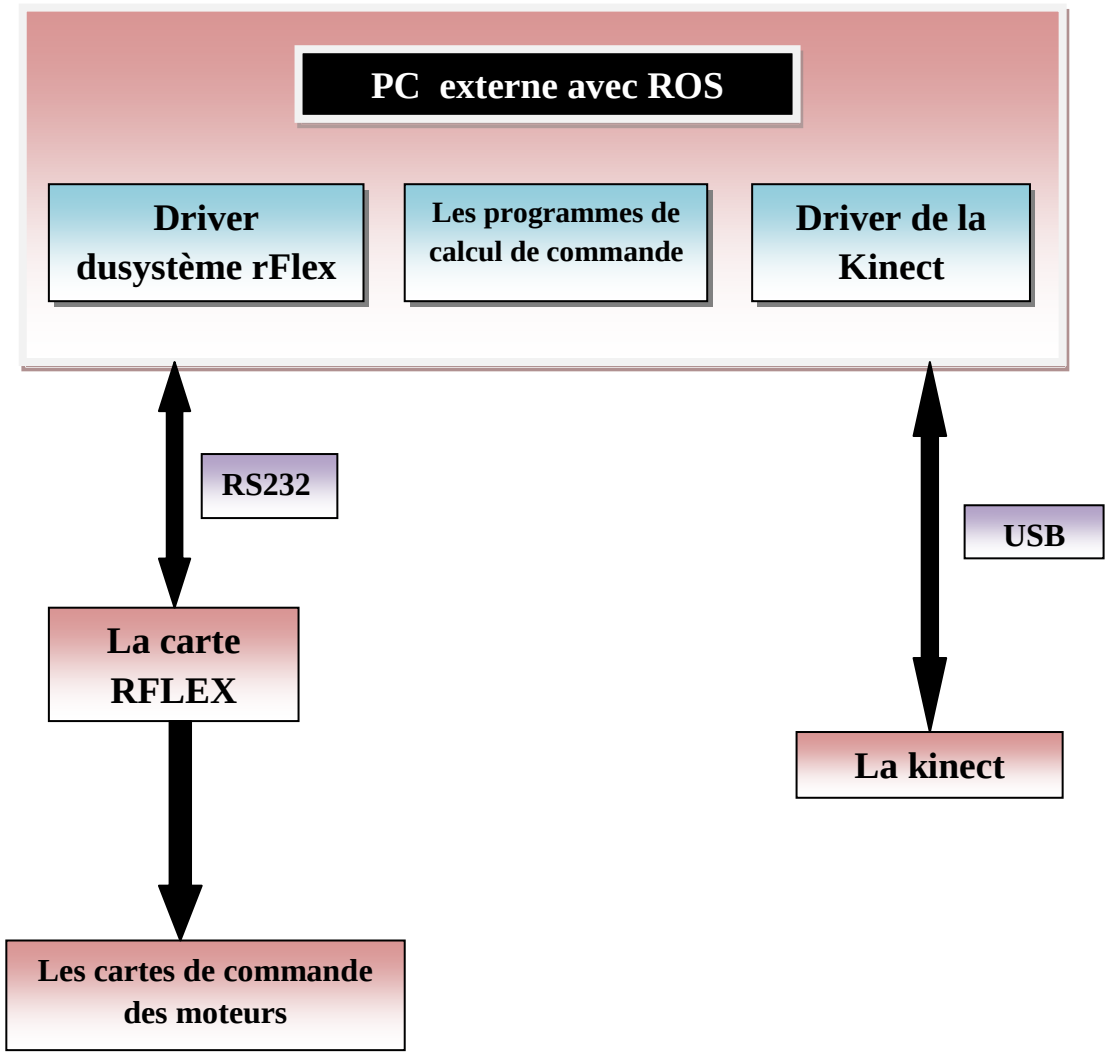


Fig.3.1 Organigramme de l'architecture globale du robot

2.1.1La Kinect

La Kinect est une caméra 3D mise au point initialement par Microsoft pour l'industrie des jeux vidéo, elle est considérée comme un multi capteurs car elle comporte trois parties :

- un projecteur de lumière infrarouge et une caméra infrarouge ;
- une camera RGB ;
- des capteurs audio.

Elle est disponible sur le marché sous différentes versions, nous avons travaillé avec **''la Kinect for Windows ''**. Cette dernière est munie d'un port **USB** classique permettant de récupérer par logicielles images en couleur et les images de profondeur afin de développer des applications se basant sur les interactions naturelles (dans notre cas, le mouvement du corps). Ceci peut être obtenu moyennant l'installation du driver open source pour la **Kinect** et du **middleware** propriétaire appelé **NITE**. Ce dernier s'occupe du traitement des images de profondeur et donc permet de segmenter les personnes en face de la caméra et, d'obtenir leur pose ou encore de reconnaître certains gestes.

2.2 Architecture software

Vue que le robot **B21r** est d'ancienne génération, sa partie software étais constituée initialement de deux parties : le système d'exportation « **Redhat** » qui est une distribution Linux et une interface « **Mobility** ».

Pour des raisons de standardisation, l'équipe robotique du **CDTA** a remplacé ce software par un outil généraliste et performant « **ROS** » qui est sollicités à travers le monde, vu le nombre de fonctionnalités et de services qu'il peut offrir.

2.3 L'outil ROS

ROS est l'abréviation de « **Robot Operating System** », comme son nom l'indique **ROS** est un système d'exploitation pour robot il a été créé en 2007 sous le nom de « ligne d'interconnexion » par le Laboratoire d'intelligence artificielle de **Stanford** avec l'appui du projet de l'AI Robot Stanford« STAIR ». En 2008, le développement s'est poursuivi principalement au **Willow Garage** (créé et financé par les dirigeants de **Google**). Cet organisme est un institut de recherche en robotique/incubateur où, plus de vingt institutions collaborent dans le sens d'un modèle de développement fédérées et ouverts (aux grandes universités et aux grands industriels de la robotique). De ce fait, l'outil **ROS** est classé open source afin d'accélérer les recherches et d'améliorer les robots.

ROS existe en deux versions la première est un système d'exploitation à part entière offrant les même services qu'un OS ordinaire cette version n'est pas disponible pour le grand public. Par contre, la deuxième version est un **middleware** devant s'interfacer avec un système d'exploitation pouvant gérer le **ROS.** Dans cette version, il existe plusieurs distributions qui ont été améliorées et dont les quatre dernières sont :

Distribution	date
ROS Indigo Igloo	Mai, 2014
ROS Hydro Medusa	Septembre, 2013
ROS Groovy Galapagos	December, 2012
ROS Fuerte Turtle	Avril,2012

Dans notre cas, nous avons travaillé avec la version **middleware** de la distribution **ROS Fuerte** sous un système d'exploitation **Linux**.

2.3.1Système de fichiers de ROS

Comme tout système d'exploitation **middleware ROS** a une structure hiérarchique pour organiser ces ressources sur le disque dont les six principaux niveaux sont :

- **le package ou Paquet logiciel :** c'est l'unité principale d'organisation logicielle de ROS. Un package est un répertoire qui contient les programmes, les exécutables, les bibliothèques externes, des données, des fichiers de configuration et un fichier de configuration xml nommé manifest.xml.
- **Manifest :** c'est un fichier qui fournit des informations concernant les packages. Il définit les dépendances et les bibliothèques utilisées par le compilateur.
- **la stack ou la Pile:** une pile est une collection de packages. Elle propose une agrégation de fonctionnalités telles que la navigation, la localisation etc. Cette pile est un répertoire qui contient les répertoires des packages ainsi qu'un fichier de configuration nommé stack.xml, c'est le « manifest de la pile ».
- **Manifest de la pile :** c'est un fichier qui fournit des informations concernant la pile (on y trouve les différents packages contenus dans la pile et d'autres information)
- **Type Message :** définit la structure de données pour les messages qui sont dans les packages.
- **Type Service :** définit la structure de données relative à la demande (requête) et à la réponse d'un service qui est stocké dans le package.

L'organigramme de la Figure 3.2 résume la structure de ce système.

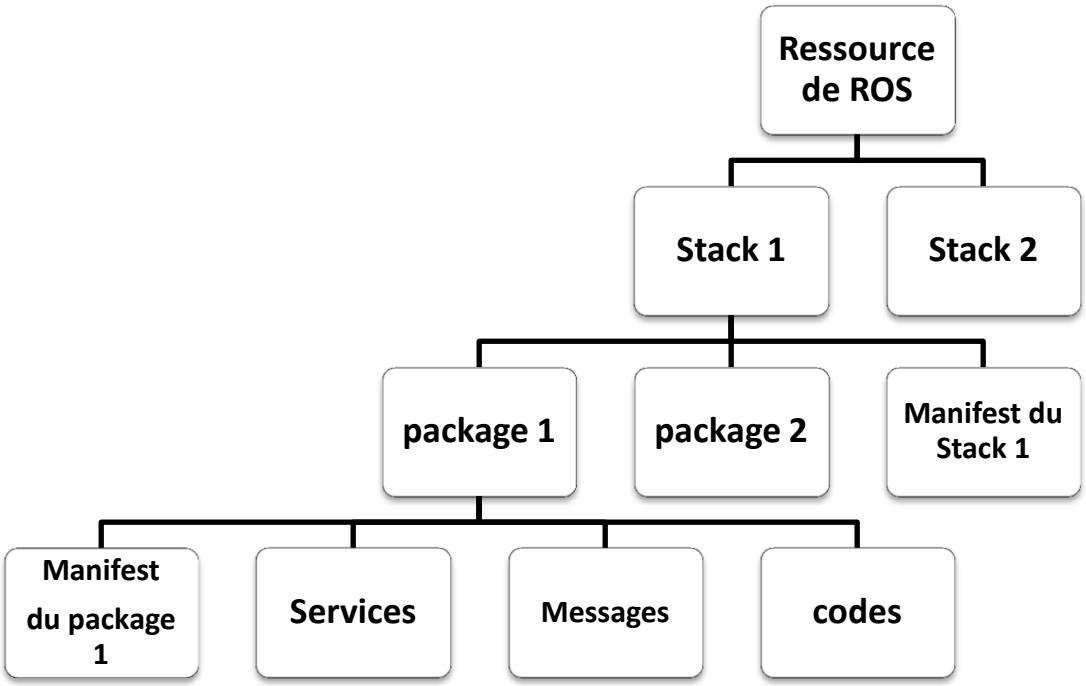


Fig.3.2 Système de fichier de ROS

2.3.2Système de traitement et calcul de ROS

En générale, l'exécution d'une tâche robotique fait appel simultanément à plusieurs exécutables car elle exploite souvent un ou plusieurs capteurs et actionneurs. Cette situation nécessité d'un système d'exploitation capable de gérer ces exécutables et d'assurer l'échange d'information entre eux soit d'une façon synchrone ou asynchrone. Pour ce faire, **ROS** intègre l'architecture suivante de traitement et de calcul :

- **Les nœuds :** un nœud est l'instance d'un exécutable développé à partir de la bibliothèque **ROS** et qui peut correspondre par exemple à un capteur ou à un algorithme de commande. L'ensemble des nœuds constituent le système de contrôle du robot.
- **Le Master:** le Master est un service de déclaration et d'enregistrement des nœuds qui permet ainsi aux nœuds de se connaître et d'échanger des données. Le Master comprend une base de données centralisée dans laquelle les nœuds peuvent stocker de l'information c'est ce qu'on appelle **'' le ParameterServer''**
- **Les topics ou les sujets :** l'échange d'information entre les nœuds peut se faire d'une façon synchrone ou asynchrone. Dans le cas où la communication est asynchrone, le

système de transport de l'information **''topic''**est basé sur le principe de la publication/l'abonnement. Ainsi, un ou plusieurs nœuds publient une donnée sous forme de message dans le **topic** et d'autres nœuds s'abonnent à ce **topic** pour récupérer cette donnée.

Les services : en mode de communication synchrone, l'échange se fait seulement entre deux nœuds où l'un d'eux envoie une requête (demandant un service) et le deuxième lui répond.

3. Programmes implémentés

Les programmes, implémentant notre tâche robotique sur le B21r, sont répartis sur trois packages : **rFlex, kinect_suivi_av**, **commande_av**. A leurs tour, ces packages contiennent des nœuds.

3.1Le package rFlex

Ce package a été développé par la communauté ROS afin d'assurer la communication entre les différents nœuds contenues dans le **PC externe** et la carte **rFlex** pour que cette dernière puisse piloter le robot. Ce package contient essentiellement deux nœuds **rFlex_driver.cpp** et **b21_node.cpp**dont la fonction est de récupérer et d'envoyer les données à différents sous-systèmes à travers plusieurs **topics**. Par exemple, le sous-système des moteurs reçoit les commandes en vitesses de rotation et de translation via le **topic cmd_vel**, et le sous-système capteurs (ces deux nœuds) récupère l'odométrie du robot et les publie dans le **topic odom**. Dans notre travail, on a utilisé le sous-système des moteurs et le topic **cmd_vel**.

3.2 Package kinect_suivi_av

Ce package traite la partie réservée à la perception, il a été créé en modifiant le package **Openni_tracker** appartenant au cadre **OpenNI** lequel fournit un ensemble d'interfaces de programmation d'applications open source pour les dispositifs physiques et plusieurs **middlewares**. Il nous a permis, dans notre cas, d'interfacer le **middleware NITE**.

Ce package contient un seul nœud **kinect_suivi1.cpp**qui estime la posture du squelette humain comme un ensemble de 15 articulations du corps humain et il fournit le suivi de ce dernier à partir des deux images RGB/Depth capturées par la Kinect.

Pour répondre aux besoins de notre application, le nœud **kinect_suivi1.cpp** publie trois messages dans trois **topics** :

- Un message qui contient les coordonnées métriques des quatres points choisis, il est publié dans le topic **Position_métrique.**
- Un message contenant les coordonnées 3D des quatre points choisis, il est publié dans le topic **Postion_3D**.
- Un message qui contient les paramètres de translation et de rotation du repère lié à l'articulation « torso » par rapport au repère Kinect exprimés en quaternions et coordonnées de l'origine ce dernier est publié dans le topic **cmd_3D**.

3.3 Package commande_av

Ce package est le cœur de notre application car il contient les trois nœuds dédiés au calcul de la commande en vitesse de l'asservissement visuel basé sur les méthodes développées au chapitre2.

- Le nœud **Commande_2D.cpp** est chargé de calculer les vitesses du robot en utilisant la loi de commande 2D dans trois cas dépendant du choix de la matrice d'interaction. Ce nœud présente un menu où l'utilisateur peut choisir la forme de la matrice d'interaction parmi trois choix proposés:
    1. Pour le 1ier choix$\boldsymbol{L} = \boldsymbol{L}(\boldsymbol{s}^*, \boldsymbol{z}^*)$: la matrice d'interaction est calculée en une seule fois pour les valeurs de consignes des indices visuels.
    2. Pour le$2^{éme}$ choix $\boldsymbol{L} = \boldsymbol{L}(\boldsymbol{s}(\boldsymbol{t}), \boldsymbol{z}(\boldsymbol{t}))$: la matrice d'interaction est déterminée à chaque instant de calcul pour la valeur courante des indices visuels.
    3. Pour le $3^{éme}$ choix $\boldsymbol{L} = (\boldsymbol{L}(\boldsymbol{s}(\boldsymbol{t}), \boldsymbol{z}(\boldsymbol{t})) + \boldsymbol{L}(\boldsymbol{s}^*, \boldsymbol{z}^*))/\mathbf{2}$, dans ce cas la matrice d'interaction est la moyenne des deux cas précédents.

Une fois le choix de la matrice d'interaction est fait le menu demande à l'utilisateur de se mettre à une distance désirée du robot pour prendre l'image consigne et lancer le suivi.

Ce nœud lit les valeurs des indices visuels à partir d'un message publié par **kinect_suivi1.cpp** dans le topic **Position_métrique** et publie les vitesses calculées dans le topic **cmd_vel** pour que ces dernières soit transmises au robot par le nœud **rFlex_driver.cpp.**

- Le nœud **Commande_3D.cpp** calcule les vitesses du robot en utilisant l'asservissement visuel 3D point développée au chapitre2. Pour cela, il utilise les

informations visuelles publiées dans le topic **Postion_3D**et il publie à son tour les vitesses de commande dans le topic **cmd_vel.**

- Le nœud **cmd_3D_retour.cpp** utilise l'asservissement visuel 3D avec un retour d'état pour calculer les vitesses du robot. Ce nœud exploite les informations visuelles publiées dans le topic **cmd_3D** et publie les vitesses de commande dans le topic **cmd_vel.**

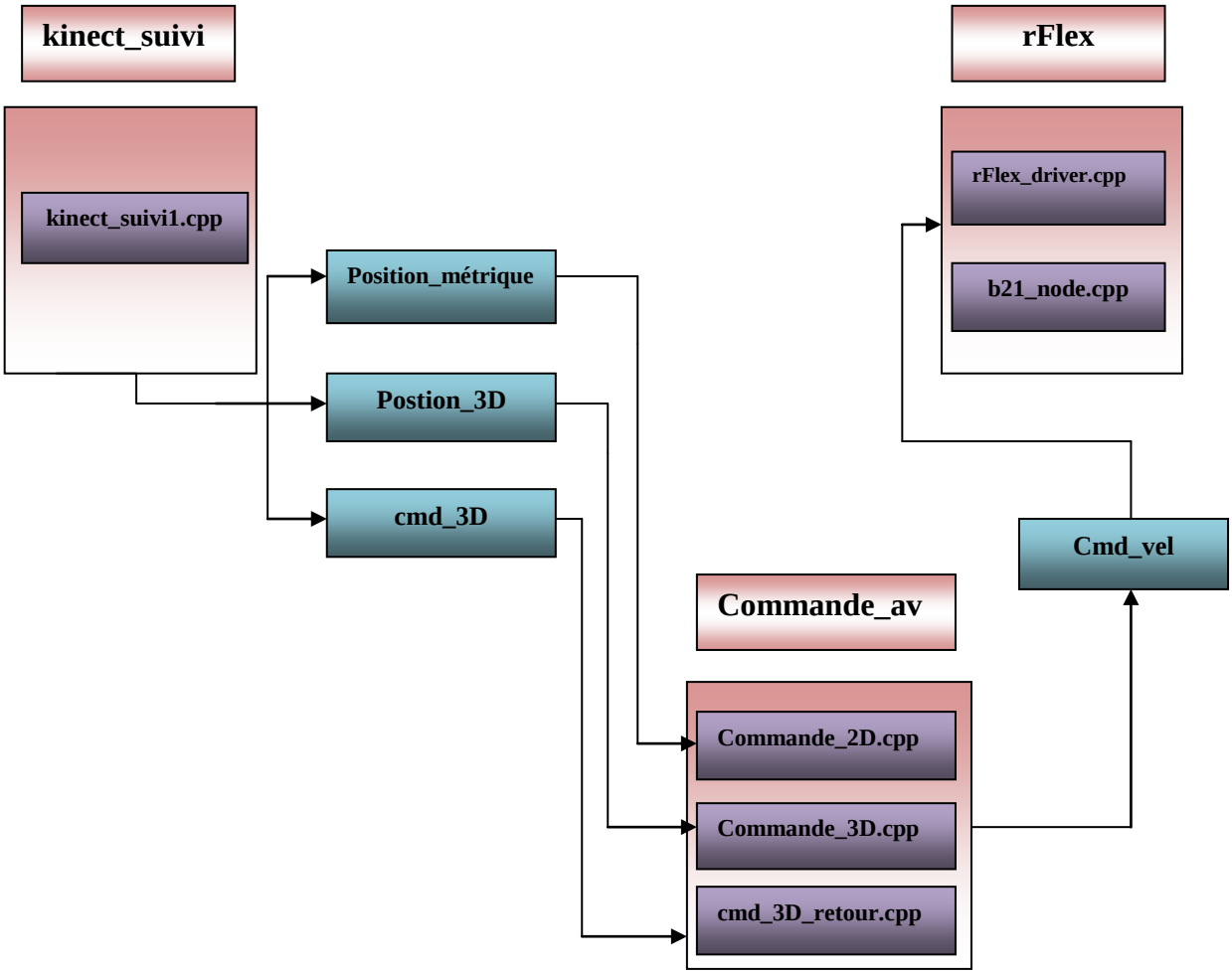


Fig.3.3 : Organisation des packages et nœuds

4. Conclusion

Dans ce chapitre, nous avons présenté l'architecture matérielle de notre système robotique c'est-à-dire le robot **B21r**équipé de la **Kinect**. Ce système est connecté à un PC externe destiné au calcul de la commande en vitesse pour l'asservissement visuel. Nous avons également présenté l'outil robotique **ROS** que nous avons installé sur le PC afin d'assurer une gestion des différents packages nécessaires à la réalisation de la tâche proposée à savoir le suivi d'une personne par notre robot.

En dernier, nous avons présenté les différents nœuds implémentés et les interconnections existantes entre eux. Ces nœuds représentent le nœud de la perception, les nœuds de calcul des différentes commandes et les nœuds driver de la carte de commande du robot.

Dans le prochain chapitre, nous allons tester et évaluer les différentes commandes développées au chapitre 2.

CHAPITRE 4

EXPERIMENTATION

1. Introduction

2. Asservissement visuel 2D

2.1 Fonction de tâche et Trajectoire des indices visuels

2.2 Influence de la matrice d'interaction sur les performances de la commande

3. Asservissement visuel 3D

3.1 Cas de la commande 3D point

3.2 Commande 3D avec retour d'état

4. Comparaison de l'asservissement visuel 2D et 3D

5. Conclusion

1. Introduction

Nous présenterons, dans cette partie, les différents tests effectués en suivant un scénario bien définie au préalable dans le but de valider les asservissements visuels proposés. Par ailleurs, ces tests ont étaient effectués sur la plateforme mobile réelle. Cependant et dans le but de préserver le matériel mis à notre disposition, nous avons, tout d'abord, procédé à une vérification de la viabilité des commandes élaborées. En effet, nous avons procédé à des tests sans la connexion du robot en lisant les vitesses publiées par le nœud de la commande pour s'assurer de l'admissibilité des vitesses envoyées au robot et de la non saturation de la commande des actionneurs.

La tache de la commande consiste en une mission de suivi d'une cible dynamique (déplacement d'une personne) au moyen d'un asservissement visuel.

A cet effet, nous exposerons, dans un premier temps, les résultats de l'asservissement 2D pour les trois choix de la matrice d'interaction afin d'en évaluer la convergence des indices visuels. On procédera de la même manière pour le cas de la commande 3D où les deux variantes (3D point et 3D par retour d'état) sont exploitées. Puis, nous analysons et comparons les performances ces asservissements visuels 2D et 3D.

2. Asservissement visuel 2D

La mission de suivi d'une cible par asservissement visuel 2D revient à faire converger les indices visuels de la cible dynamique dans l'image vers les indices visuels de référence introduits au préalable reflétant la position désirée entre le robot et la cible. Dans notre cas nous avons sélectionnés quarts points relatifs à la tête, au torse et aux deux épaules.

Nous présentons, en premier lieu, les résultats obtenus en utilisant la loi de commande (2.54)et en fonction du choix de la matrice d'interaction pour une matrice diagonale des gains telle que : $\lambda 11 = \lambda 22 = \lambda 33$=0.08. Il est important de noter que ces gains n'ont été fixés à cette valeur finale qu'après plusieurs tests d'essais-erreurs sans la connexion du robot. La détermination de la matrice d'interaction peut être effectuée de trois manières suivantes :

- Cas1 d'une matrice d'interaction courante $\boldsymbol{L(s,z)}$ **:** elle est calculée en ligne en fonction de la mesure de profondeur et des indices visuels issus de la Kinect, cette matrice est mise à jour à chaque itération.
- Cas2 d'une matrice d'interaction constante $\boldsymbol{L(s,z)} = \boldsymbol{L(s^*,z^*)}$ **:** elle est calculée hors ligne en fonction de la profondeur et des primitives visuelles relatives à la position de référence.
- Cas3 d'une matrice d'interaction moyenne (hybride) $\boldsymbol{L(s,z) = \left[L(s(t),z(t)) + L(s^*,z^*)\right]/2}$ **:** elle est prise comme étant la moyenne entre la matrice courante et celle constante, donc elle aussi est déterminée à chaque itération.

2.1 Fonction de tâche et Trajectoire des indices visuels

Le test du système robotisé est effectué pour un scénario bien défini. Pour ce scénario, tout d'abord, le système robotisé prend une photo de la cible à la position de référence par rapport au robot par la suite la perception est stoppée le temps nécessaire à la cible de se déplacer vers une autre position que celle désirée. Une fois la cible est en place, la mission de détection du robot est mise en route par l'intervention de l'asservissement visuel.

Pour les trois variantes de calcul de la fonction de la tâche, les **Figure4.1** à **4.3** donnent la trajectoire des primitives dans le plan image et l'évolution de la fonction de la tâche considérée.

Nous analysons en premier la fonction de la tâche et la trajectoire de convergence des primitives visuelles.

- Pour le cas 1,une composante de la fonction de la tâche ne converge pas vers zéro ceci signifie que celle-ci a atteint un minimum local dans le plan de l'image. Sachant que e=C(s-s*) par conséquent cela se traduit par la non convergence des indices visuels car le cadre final (formant les quatre points sélectionnées sur la cible) n'est pas superposable sur le cadre consigne. Cependant, la réalisation de la tâche robotique n'a été pas affectée. En effet, le robot a avancé vers la cible en respectant la distance voulue puisque le cadre final a pris une taille s'approchant de la taille du cadre consigne.

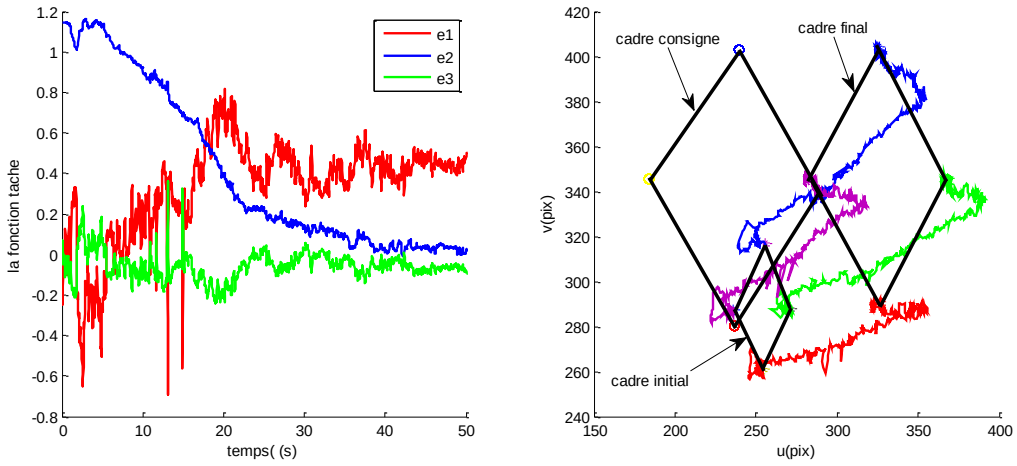


Fig.4.1 Fonction de tâche et Indices visuels dans l'image pour le Cas1

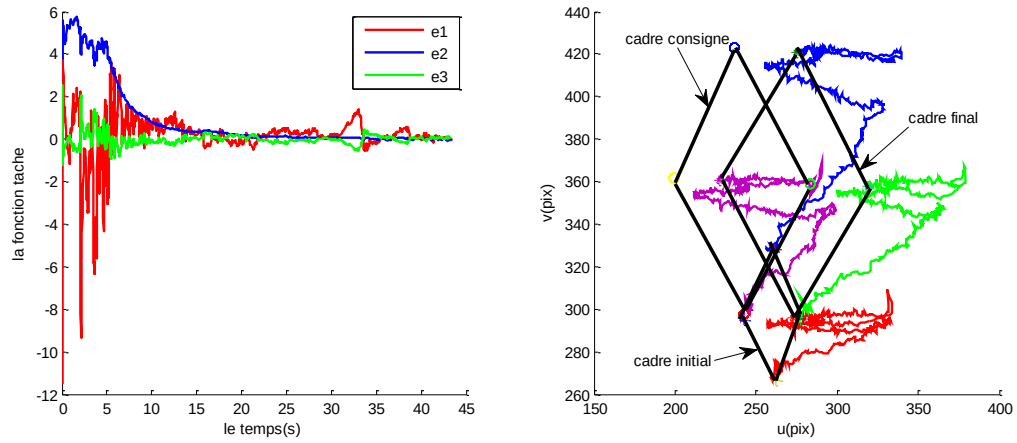


Fig.4.2 Fonction de tache et Indices visuels dans l'image pour le cas2

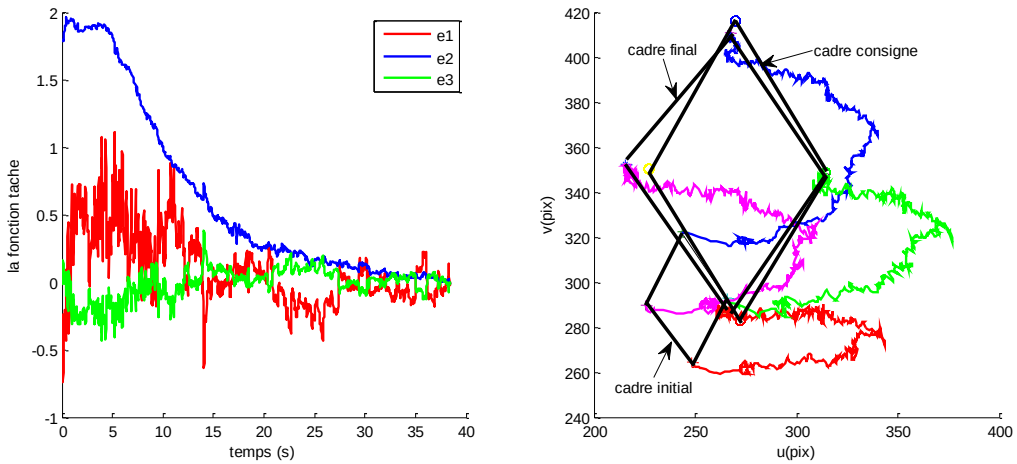


Fig.4.3 Fonction de tache et Indices visuels dans l'image pour le cas3

Durant cette mission de suivi, les indices visuels demeurent dans l'image et leurs trajectoires peuvent être assimilées globalement à une droite.

- Pour le cas 2,la fonction de tâche converge vers zéro et les trajectoires des indices visuels ne sont pas contraintes à suivre une allure de droite mais celles-ci restent dans l'image de plus, le cadre final approche la forme du cadre consigne.
- Pour le cas3, la fonction de tâche converge vers zéro et que les indices visuels coïncident presque avec les consignes. De plus, ils suivent des trajectoires en forme de boucle et sont moins complexes que ceux du 2^{éme}cas.

Ces trois tests permettent de confirmer que :

a. En asservissement 2D, la convergence de la fonction de la tâche ne conduit pas nécessairement à la convergence des indices visuels. En effet, **e=C(s-s*)**et sa convergence assure la converge de **Cs** vers **Cs***. De ce fait, la convergence des indices visuels dépend non seulement de la convergence de la fonction de tâche et mais aussi de la matrice **C**.
b. La trajectoire et la convergence des indices visuels dépendent indirectement du choix de la matrice d'interaction (**C =L(s,z)**$^{+}$). Ainsi, nous avons relevé que dans le cas3, les indices visuels références étaient bien approchés contrairement aux deux autres cas.

2.2 Influence de la matrice d'interaction sur les performances de la commande

Pour le scénario prévu et les trois variantes selon le choix de la matrice d'interaction, nous relevons l'évolution des signaux de commande du système robotisé. Nous rappelons que l signal de commande du robot est représentée par le vecteur vitesse $\dot{\boldsymbol{q}}$ lequel est constitué de la vitesse de translation dans le plan et de la vitesse de rotation autour de son axe vertical**.** Ce relevé est effectué dans le but de procéder par la suite à une comparaison de la performance de la commande selon le choix de la matrice à travers l'analyse du temps de réponse et la vitesse de convergence relative à la vitesse de rotation et à celle de translation.

Les **Figures 4.4** à **4.6** donnent l'évolution du vecteur commande pour les trois variantes de la matrice d'interconnexion. Rappelons que la vitesse de décroissance de la fonction de tâche est imposée via le choix de la matrice diagonale des gains λ où $\lambda_{11}=\lambda_{22}=\lambda_{33}=0{,}08$.

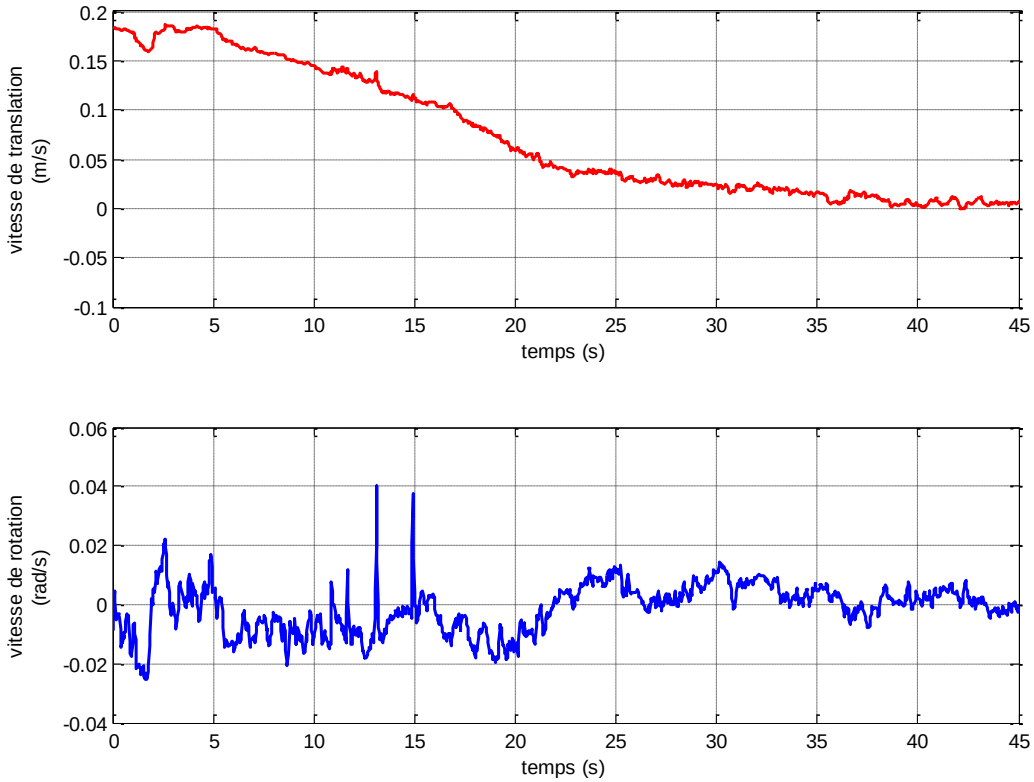


Fig.4.4 Commandes pour le cas1

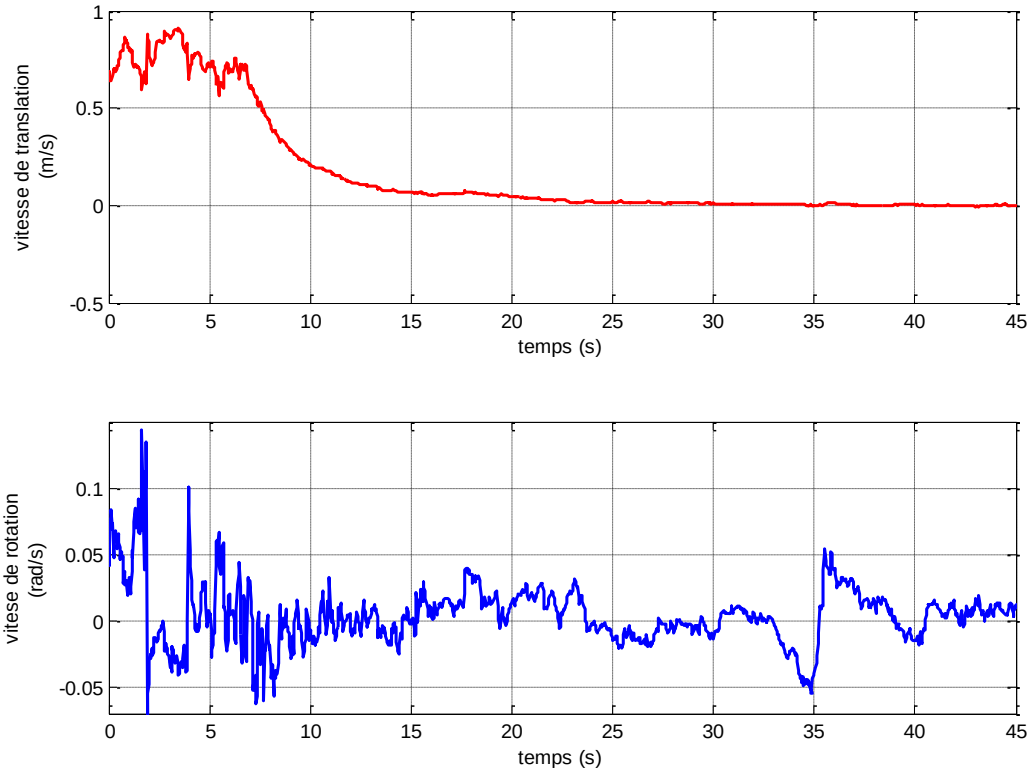


Fig.4.5 Commandes pour le cas2

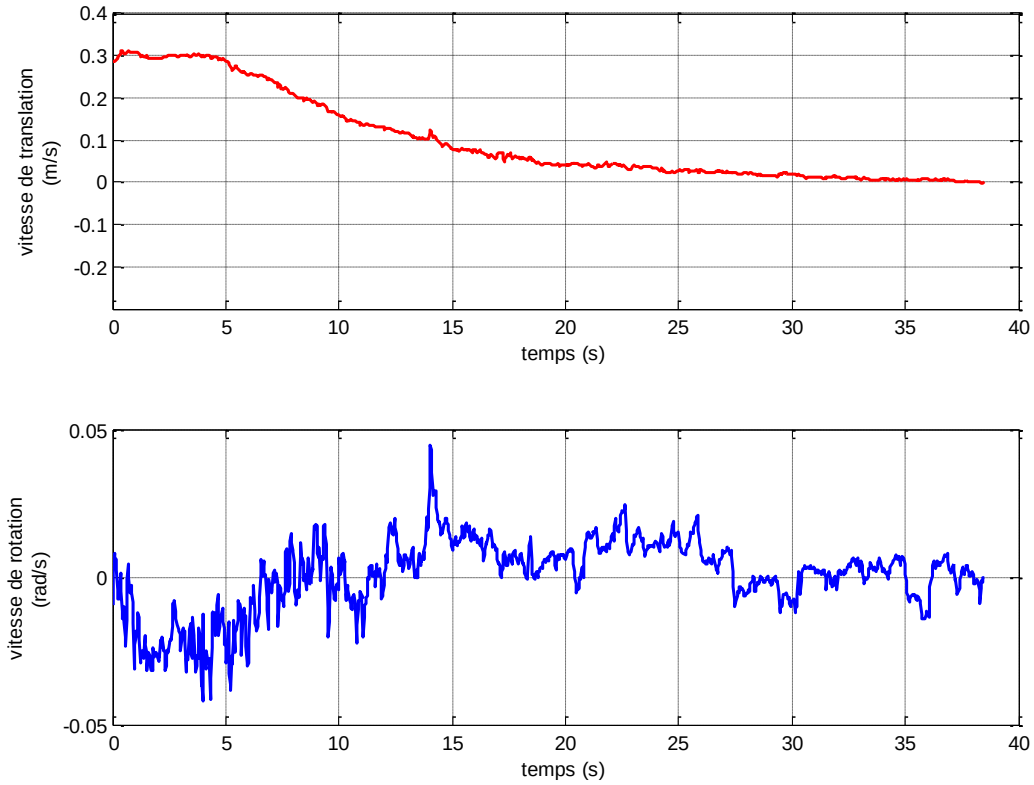


Fig.4.6 Commandes pour le cas3

Dans tous les cas, le schéma de commande tente d'assurer une décroissance en exponentiel des composantes de la fonction tâche, cependant cette dernière peut être sujette aux minimas locaux. Suivant le scénario prévu, au début (t<10s) la vitesse est importante, car l'écart **e** est d'autant plus important que la cible est loin du robot, puis plus le robot s'approche de la cible plus la vitesse décroit jusqu'à son annulation. Ceci se manifeste par l'arrêt du robot même si dans le cas1, les composantes de la fonction de tâche n'ont pas toutes été amenées à zéro et les indices visuels finaux n'ont pas totalement coïncidé avec ceux de la référence.

Concernant le temps de réponse, nous relevons un temps de convergence de 42s pour le cas1, il augmente à 45s pour le cas2 et enfin ce temps de réponse se réduit à 32s pour le cas3.

De même, nous remarquons que les commandes les plus lisses et moins ondulées concernent celles du cas3 (matrice d'interaction hybride) ceci est encore beaucoup plus valide pour la rotation.

Donc, la commande présentant les meilleures performances est celle utilisant la matrice d'interaction moyenne (hybride cas 3).

3. Asservissement visuel 3D

Contrairement à l'asservissement 2D, l'asservissement 3D opère dans l'espace capteur afin d'atteindre une position relative désirée entre le robot et la cible. Pour ce faire, nous avons implémenté deux lois de commande (2.72& 2.86). La première, appelée « 3D point », a pour objectif d'annuler l'erreur entre les coordonnées 3D des quatre points sélectionnés de la cible et ceux des points références. Tandis que la deuxième cherche à faire converger le repère de la Kinect vers un repère désiré déduit à partir de la pose désirée de la cible. Dans ce qui suit, nous allons présenter les résultats de ces deux commandes afin et d'en analyser leurs performances et de les comparer.

3.1 Cas de la commande 3D point

Pour le même scénario, l'évolution de la fonction tâche pour un asservissement 3D point apparait à la Figure 4.7. De cette figure, nous relevons qu'une des trois composantes de la fonction de tâche ne s'est pas annulée ce qui peut signifier que la fonction tâche a atteint un minimum local. Cette situation est relative à une non-convergence des indices visuels, sans pour autant affecter la tâche robotique (le suivi d'une personne) car le robot a bien suivi la cible mais l'erreur des indices visuels n'a pas convergé vers zéro.

Ce résultat peut s'explique par le fait que la commande visuelle 3D point se base sur le formalisme de la fonction tâche faisant appel à la matrice C. En effet, cette matrice peut perturber la convergence des indices visuels et rend la fonction tâche sujette aux minimas locaux.

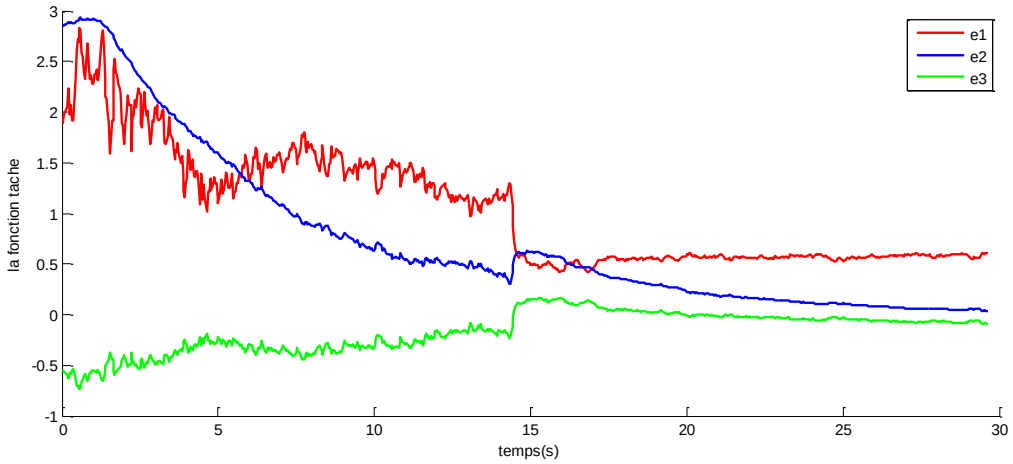


Fig.4.7 Composantes de la fonction tâche pour la commande 3D point

3.2 Commande 3D avec retour d'état

Toujours pour le cas du même scénario, la mission de suivi d'une personne est effectuée cette fois ci en implémentant l'asservissement visuel 3D basé sur le retour d'état (loi 2.86). Pour juger de la convergence de la commande, nous présentons tout d'abord, l'évolution des composantes du vecteur d'état.

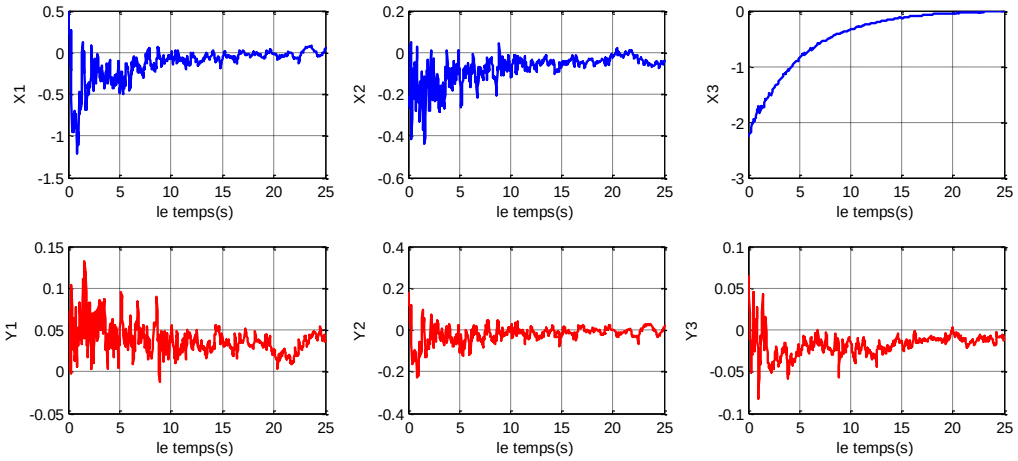


Fig.4.8 Composantes du vecteur d'état

Dans le cas de l'asservissement visuel 3D basé sur le retour d'état, la **Figure 4.8** montre l'évolution du vecteur d'état. Nous notons, la convergence de toutes les composantes du vecteur d'état, représentant la position et l'orientation du repère de la Kinect dans le repère désiré, ce qui assure la réalisation de la tache robotique de suivi d'une personne.

3.3 Performances des commandes visuels 3D points et 3D par retour d'état

Nous analysons l'évolution des commandes dans le cas de l'asservissement visuel 3D point et 3D par retour d'état afin d'établir leurs performances et de les comparer. Cette analyse se base sur les **Figures 4.9 & 4.10**. De ces figures, nous relevons que :

- Le temps de réponse la 1ére méthode est en moyenne de 25s alors celui de la seconde méthode se réduit à 20 s.

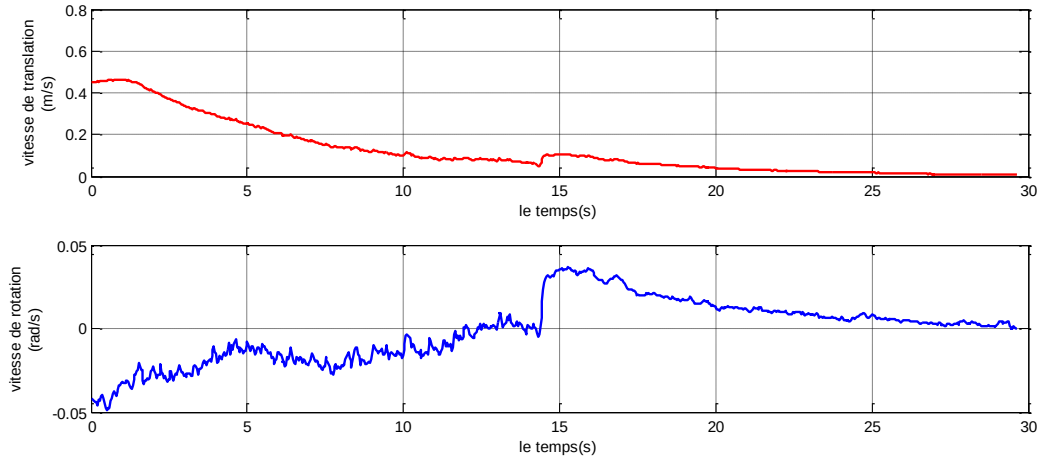


Fig.4.9 Evolution des commandes de l'asservissement visuel 3D point

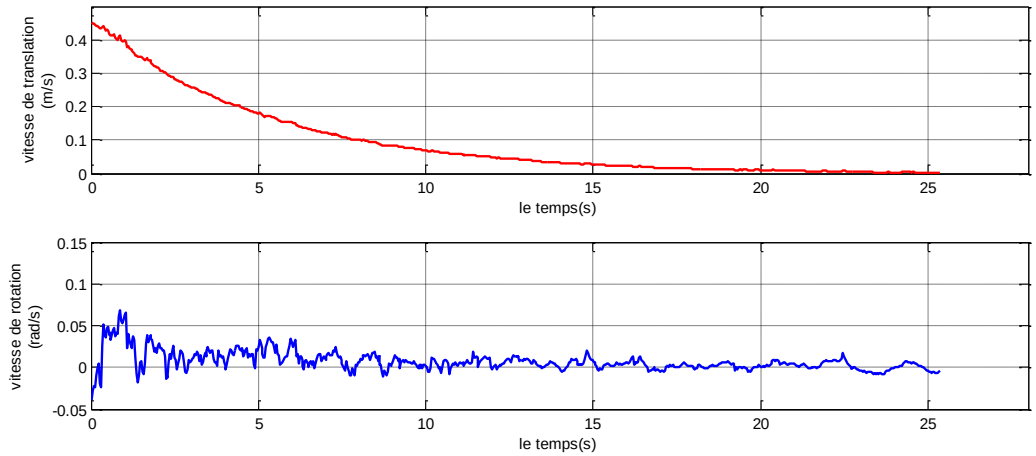


Fig.4.10 Evolution des commandes de l'asservissement visuel 3D par retour d'état

- La commande 3D par retour d'état est bien meilleure car elle est moins broutée par rapport à celle du 3D point et de plus elle est plus rapide. En outre, elle ne fait pas intervenir la matrice de combinaison C en conséquence elle est moins sujette au problème des minimas locaux.

4. Comparaison de l'asservissement visuel 2D et 3D

Il est clair que l'objectif de commande est la poursuite d'une cible, en exploitant soit les informations 2D dans le plan image soit, les informations 3D dans l'espace capteur. La comparaison est menée sur la base des résultats relatifs au cas3 de l'asservissement 2D et à la commande 3D par retour d'état. Ces résultats montrent une convergence de tous les indices

visuels pour le cas la commande 3D alors que dans la cas l'asservissement 2D la convergence des indices visuels n'est pas assurée même si la tâche est bien régulée. En effet, la commande 2D est fondée sur le formalisme de la fonction tâche qui fait appelle à une matrice de combinaison C où le but de la commande est d'amener vers zéros la fonction tache ***e=C(s-s*).*** Ainsi, la convergence de ***e(t)*** assure la convergence de ***Cs(q,t)*** vers ***Cs*(q,t)***non celle de ***s(q,t)*** vers ***s*(t).***

Par ailleurs et lors des tests, nous avons aussi remarqué que dans le cas de l'asservissement 2D et pour la situation où la cible reste dans la position désirée mais se met de profil par rapport au robot, celui avance brusquement vers elle. Ceci s'explique par le fait que les deux points des épaules se rapprochent dans l'image et correspondent à la situation où la cible est loin du robot ce qui empêche la réalisation de la tâche. Par contre, dans cette même situation de la cible par rapport au robot, l'asservissement 3D avec retour d'état permet au robot de se déplacer pour se mettre en face de la cible.

Le seul problème rencontré en 3D est lié à la perception dans la situation où la personne se retourne le rebot se déplace pour se mettre en face et durant son déplacement il perd sa cible car elle n'est plus dans son champs de vision.

5. Conclusion

Dans ce dernier chapitre, après avoir défini le scénario suivi, nous avons expérimenté les lois d'asservissements visuels synthétisées au chapitre 2 sur la plate-forme robotique. Cette expérimentation a été effectuée en trois parties.

En première partie, nous avons testé les trois cas de l'asservissement 2D, les résultats obtenus pour le même scénario nous ont permis d'établir que l'asservissement 2D basé sur la matrice d'interaction moyenne (hybride) a conduit à meilleures performances. En effet, les indices visuels de références étaient bien approchés contrairement aux deux autres et que les commandes sont plus lisses et moins broutées. De plus, nous avons pu observer que la convergence de la fonction de tâche n'implique pas forcément la convergence des indices visuels.

En deuxième partie, nous avons testé les deux méthodes (3D point et 3D par retour d'état). La commande 3D point a conduit la fonction de tâche à un minimum locale et donc à une

non-convergence des indices visuels sans affecter la tâche robotique de suivi. Par contre, la 2^{ième} méthode a permis à toutes les composantes du vecteur d'état de converger pratiquement vers zéro ce qui implique la réalisation de la tâche robotique de suivi. De plus, la dynamique de la commande 3D par retour d'état est bien plus rapide.

Dans la troisième partie, nous avons établi une comparaison entre le cas par matrice d'interaction moyenne de l'asservissement visuel 2D et l'asservissement 3D par retour d'état. Celle-ci, nous a permis de conclure que l'asservissement 3D par retour d'état a conduit à meilleures performances pour notre application.

CONCLUSION GENERALE

Dans le cadre de ce mémoire, nous avons traité le problème de la navigation d'un robot guide basée sur un capteur visuel. Pour ce faire, nous avons tout d'abord présenté le système robotique destiné à notre application : le robot B21r équipé d'un capteur de vision (Kinect). Puis, nous avons enchainé par un bref rappel théorique relatif aux techniques d'asservissements visuels les plus connues et par une modélisation fondée sur le concept des transformations homogènes de l'ensemble robot-caméra-cible.

Le problème de l'asservissement visuel, proprement dit, est abordé sur la base du formalisme de la fonction de tâche. Dans ce contexte, nous avons alors proposé des lois d'asservissement visuel du type 2D et du type 3D. Le type 2D exploite les informations visuelles extraites de l'image où nous avons traité trois variantes en fonction de la matrice d'interaction. Pour le type 3D, nous avons exploré deux variantes : la première (3D points) assure la convergence des points 3D de la cible vers des points désirés et la deuxième (3D par retour d'état), étant une commande 3D, la cible est considérée par son modèle tridimensionnel situé dans le repère du capteur. Les lois de commandes synthétisées sont compatibles avec la tâche considérée dans notre étude, à savoir : « le suivi d'une personne dans un milieu intérieur ».

Ensuite, nous avons fixé un même scénario pour expérimenter, sur la plateforme robotique, les différentes lois d'asservissement visuel. Les résultats obtenus de cette expérimentation nous ont permis de relever que :

- Pour l'asservissement 2D, la convergence de la fonction de tâche n'assure pas forcément la convergence des indices visuels. De plus, le choix d'une matrice d'interaction hybride permet aux indices visuels de mieux converger vers les indices visuels références et aux commandes associées d'être les plus lisses et les moins broutées.
- La fonction de tâche associée à la commande 3D points présente un minimum locale par conséquent la non convergence des indices visuels, ceci sans affecter la tache robotique du suivi puisque le robot a bien suivi la cible.

- Dans le cas de la commande 3D par retour d'état, les composantes du vecteur d'état convergent pratiquement toutes vers zéro ce qui assure la réalisation de la tache robotique considérée.

Ces résultats expérimentaux, nous ont également permis d'effectuer une étude comparative des performances des lois d'asservissement visuel implémentées. Il en ressort que la commande 3D par retour d'état présente les meilleures performances parmi toutes les autres commandes.

D'après cette courte expérience, et à travers les résultats des tests effectués, il nous semble intéressant de continuer ce travail en traitant par la suite les thèmes suivants :

- Palier à la perte de la cible pour le cas des commandes 3D en équipant le robot d'une PTU, ou utiliser les indices estimés.

- Intégrer des identifiants de personnes (ex : reconnaissance de visage).

- Introduire un évitement d'obstacles pour plus d'autonomie.

Annexe A : Détection et suivi de l'être humain

- La Librairie OPENNI a déposé une application sous Linux qui permet de détecter les personnes sur la scène (elle détecte 4 personnes, et permet le suivi de 2 personnes) . Cette application permet de visualiser l'utilisateur comme un squelette,Le système de base et les algorithmes de cette application qui ont été développés par par A. Karsont décrits ci-dessous : Les entrées de l'algorithme sont les données RGB et les données de profondeur issus de la Kinect. Les données en sortie sont la silhouette de l'être humain représentée par les 8 points de ses articulations.

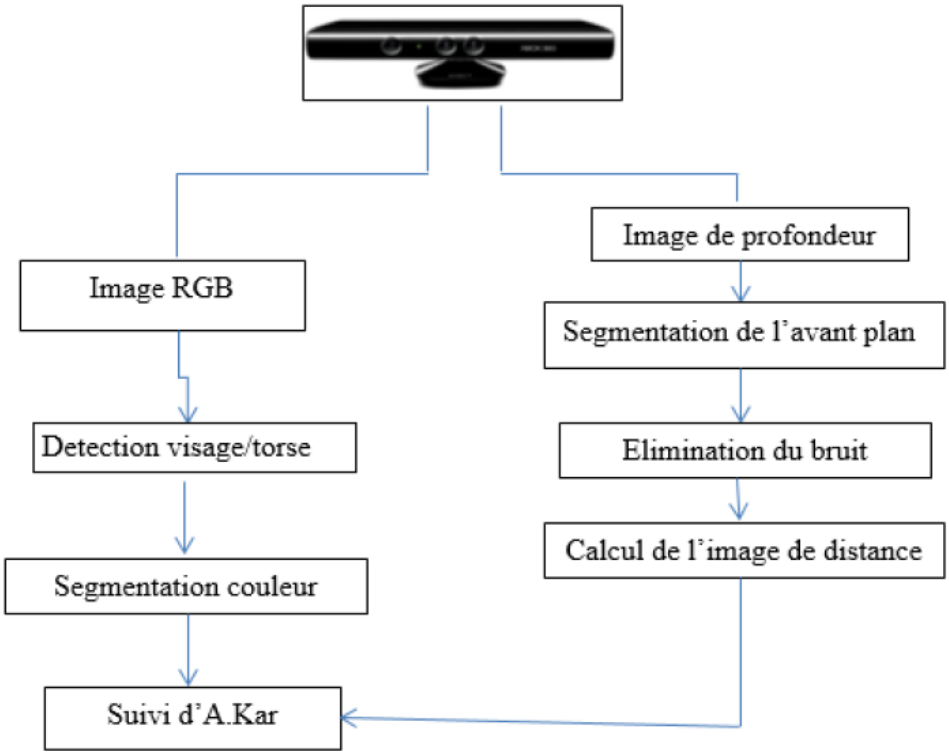
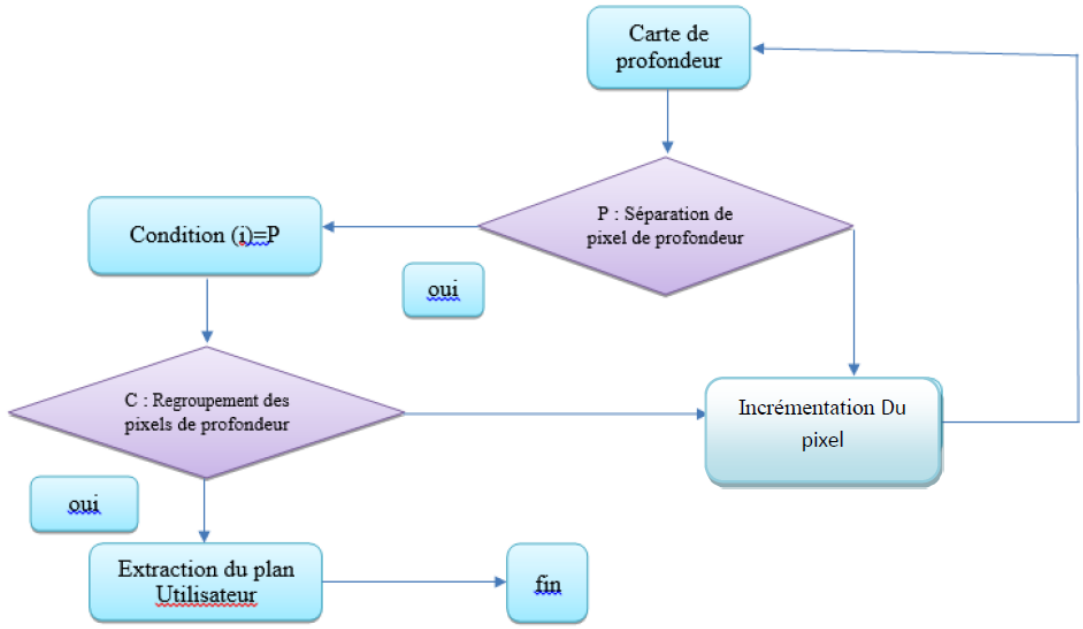


Diagramme de la méthode A. Kar pour le suivi de l'être humain

✓ **Pour L'Image de profondeur :**

Sur l'image de profondeur l'algorithme procède à la séparation entre l'avant plan et le fond, ensuite la détection de l'être humain se fait par seuillage ou par étiquetage de la carte de profondeur par l'algorithme des k-moyens selon l'organigramme donné à la figure.



Organigramme de Segmentation Avant-plan

Où P représente le changement d'intensité entre deux pixels par rapport à un seuil, C représente le regroupement entre deux pixels par rapport à un seuil. Les seuils sont récupérés à partir de la carte de profondeur.

Dans la Séparation des pixels de profondeur après récupération de la carte de profondeur, la vérification de grands changements d'intensité des pixels a été faite pour coordonner dans l'avant plan / arrière-plan de l'être humain. Après cette étape le regroupement des pixels a été fait pour extraire le plan de l'être humain, à la fin de cette opération on obtient un avant-plan utilisateur.

Segmentation l'avant-plan/arrière-plan

Puis il procède a :

-L'Elimination des bruits : Cette partie est utilisée pour éliminer les bruits qui apparaissent sur le plan utilisateur après la segmentation de l'avant-plan/arrière-plan, pour cela des opérateurs morphologiques : érosion et dilatation ce qui nous permettra de focalisé uniquement sur l'être humain.

-Le Calcul de l'image de distance **:** qui est uneTransformation appliquée sur l'image de profondeur segmentée (avant plan), pour estimer la silhouette de l'être humain.

Image de la distance

- ✓ **Pour L' image RGB :**Les étapes sont les suivantes

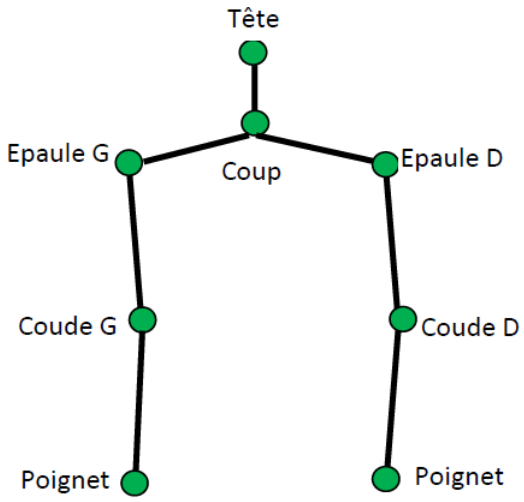
-Détection visage /torse **:** L'algorithme de classification Haar-carscade-classifier est utilisé, pour détecter le torse ensuite elle détecte le visage de l'être humain, une règle de rejet est définie cette dernière permettant de dire si une région est un objet d'intérêt ou pas.

-Segmentation couleur **:** pour pouvoir détecter la peau, l'image RGB est projetée dans l'espace de couleur HVS ce qui donne un masque binaire avec les régions de la peau.

Segmentation de la peau

- ✓ **Pour Le Suivi d'A. Kar :**

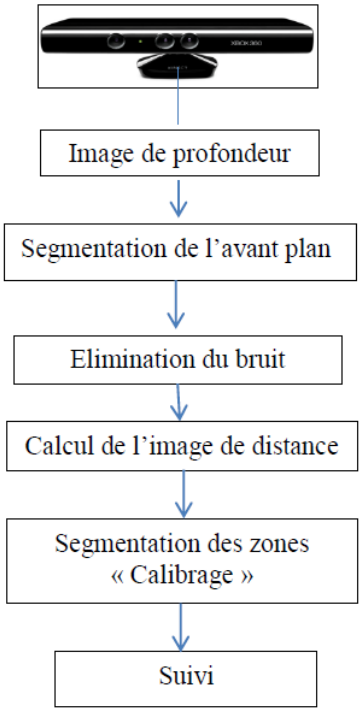
Dans cette étape, un modèle du squelette de l'être humain est utilisé, il est composé de 8 points, ensuite le suivi du squelette se fait en utilisant les images de distance et les images de la peau segmentées.



Modèle du squelette de l'être humain A.Kar

Pour faire une mise en correspondance entre le squelette de l'être humain et l'image de distance, des suppositions sont faites sur les positions initiales de la tête et du coup, tel que la tête soit au centre du rectangle de détection du visage et le coup soit au milieu du coté bas du rectangle, de même pour les épaules, sont fixés au milieu des rectangles de détection du visage et du torse.
L'estimation du coude se fait par la recherche dans un angle de pivot de l'épaule, de même pour l'estimation du poignet qui se fait dans un angle de pivot du coude.

La figure suivante présente un schéma global de la méthode A.Kar pour le suivi de tout le corps humain à partir de la Kinect :

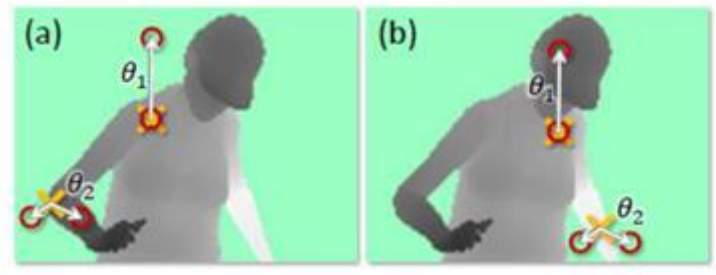


Modèle du squelette de l'être humain

En appliquant cette Méthode à notre situation :

Dans notre cas le capteur de profondeur est la Kinect, les étapes pour suivre et détecter le visiteur dans la scène par notre robot sont : en premier la détection par le capteur de profondeur (Segmentation du l'avant plan, éliminations du bruit et calcul de l'image de distance) puis une segmentation des zones pour identifier 31 zones du corps humain. Après cette identification le suivi s'applique sur un modèle de squelette.

-Segmentation des zones **:**Dans cette étape, une mesure de décalage des pixels est appliquée pour classifier le corps humain de manière à identifier jusqu'à 31 zones du corps .



Caractéristique de profondeur de pixel dans l'image

Dans la figure les cercles rouges indiquent les pixels de décalage. En (a), les deux dispositifs d'exemple donnent une réponse importante de la différence de profondeur. En (b), les deux mêmes caractéristiques à de nouveaux emplacements d'image donnent une réponse beaucoup plus petite.

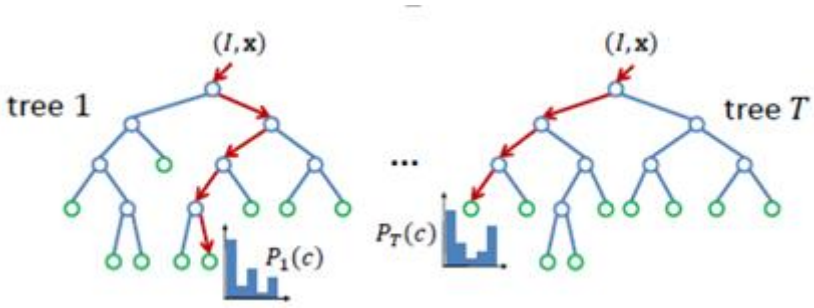
Ce décalage de pixel et mesuré par la relation suivante :

$$f_\vartheta(\mathrm{I},\mathrm{x}) = d_l\left(x + \frac{u}{d_l(x)}\right) - d_l\left(x + \frac{v}{d_l(x)}\right) \quad (\mathrm{I.1})$$

Où : $d_l(x)$est la profondeur du pixel (distance) x dans l'image I,

et les paramètres θ = (u,v) décrivent les décalages u et v.

Le calcul du décalage de la profondeur obtenu après la segmentation et l'élimination des bruits à l'avant -plan / arrière-plan, pour identifier les zones du corps à chaque instant t, ceci est difficile et très lent, pour cela un classificateur est intégré pour améliorer la rapidité de calcul et l'efficacité, l'arbre de décisions randomisés (Randomized decision forests) est appliqué . Cet arbre est une classificatrice multi-classes rapide et efficace pour de nombreuses tâches , comme l'exemple d'arbres suivant .



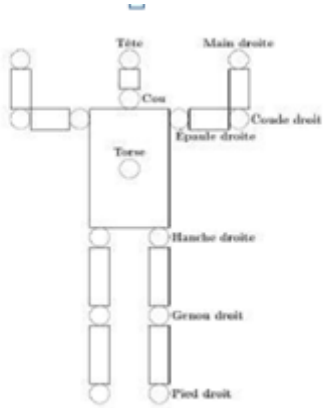
Ensemble d'arbre

Une forêt est un ensemble d'arbres de décision T, chacun étant constitué de nœuds et de feuilles fendues. Chaque nœud de répartition se compose d'une caractéristique f_ϑ et un seuil τ. Pour classer le pixel x dans l'image I, on commence à la racine et évalue l'équation (I.1) à plusieurs reprises intégré à gauche ou à droite en fonction de la comparaison de seuil τ . Au nœud feuille obtenu dans l'arbre t, de distribution appris Pt (c| I,x) sur une partie du corps étiquettes c est stockée. Les distributions sont moyennées pour tous les arbres dans la forêt de donner le classement final

P (c | I,)=$\frac{1}{T}\sum_{t=1}^{T}$ Pt (c | I, x)(I.2)

Rappelons que chaque arbre utilise l'algorithme *Randomizedtrees*. On appelle cette étape par le calibrage de l'être humain.

Apres avoir assuré le calibrage on applique un modèle de squelette composé de 15 points sur des zones segmentée du corps humain tel que chaque nœud de ce squelette correspond à une partie de la zone.



Modèle du squelette humain

Ainsi est assuré la détection et suivi de l'être humainà partir de la Kinect en utilisant la librairie OPENNI .

Annexe B : Eléments de la robotique

1. Matrice de rotation autour d'un axe u quelconque :

Soit un vecteur unitaire $\underline{\boldsymbol{u}}$ quelconque. Soit $\boldsymbol{R_i}$, un repère tel que son origine coïncide avec celle du vecteur $\underline{\boldsymbol{u}}$. Les coordonnées de $\underline{\boldsymbol{u}}$ sont alors, dans $\boldsymbol{R_i}$: $\underline{\boldsymbol{u}} = (u_X, u_Y, u_Z)^T$.

Soit $\boldsymbol{R_j}$, un repère tel que :

– son origine coïncide avec celle du vecteur $\underline{\boldsymbol{u}}$,

– son axe $\boldsymbol{Z_j}$ se confonde avec le vecteur $\underline{\boldsymbol{u}}$.

La transformation homogène permettant le passage du repère $\boldsymbol{R_i}$ au repère$\boldsymbol{R_j}$ peut être décomposée en :

– $\boldsymbol{Rot(\underline{z}, \alpha)}$: une rotation autour de l'axe $\underline{\boldsymbol{z}}$ d'un angle $\boldsymbol{\alpha}$ tel que u ∈ au plan $\boldsymbol{(z_f , y_f)}$,

– $\boldsymbol{Rot(\underline{x}, \beta)}$: une rotation autour de l'axe $\underline{\boldsymbol{x}}$ d'un angle **β**.

On obtient la relation suivante :

$$\boldsymbol{T_j^i = Rot(z, \alpha).Rot(x, \beta)} \qquad \textbf{(1.1)}$$

Cette relation est illustrée sur la figure1 Le repère R_i subit donc deux transformations successives:

$\boldsymbol{R_i}$ + $\boldsymbol{Rot(z, \alpha)}$==> $\boldsymbol{R_f}$ + $\boldsymbol{Rot(x, \beta)}$) ==> $\boldsymbol{R_j}$

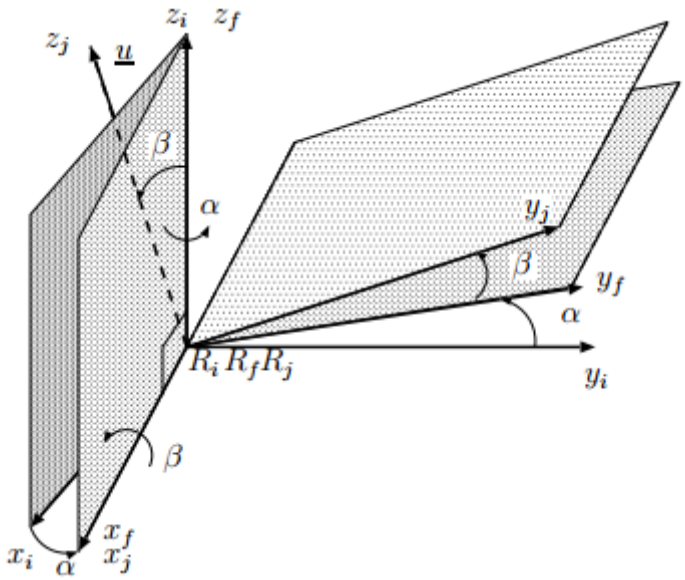


Fig 1 : rotation du repère $\mathbf{R_i}$ autour d'un axe $\underline{\boldsymbol{u}}$

En développant la relation $\boldsymbol{T_j^i = Rot(z,\alpha).Rot(x,\beta)}$,on obtient :

$$\begin{pmatrix} c\alpha & -s\alpha & 0 & 0 \\ s\alpha & c\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\beta & -s\beta & 0 \\ 0 & s\beta & c\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{c\alpha} & \boldsymbol{-s\alpha c\beta} & \boldsymbol{s\alpha s\beta} & \mathbf{0} \\ \boldsymbol{s\alpha} & \boldsymbol{c\alpha c\beta} & \boldsymbol{-c\alpha s\beta} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{s\beta} & \boldsymbol{c\beta} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \quad \textbf{(1.2)}$$

A partir de la troisième colonne de la matrice $\boldsymbol{T_j^i}$, on peut donc extraire les coordonnées du vecteur $\underline{\boldsymbol{u}}$, exprimées dans le repère$\mathbf{R_i}$:

$$\underline{u} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \begin{pmatrix} s\alpha s\beta \\ -c\alpha s\beta \\ c\beta \end{pmatrix} = a_j^i \quad (1.3)$$

Tourner autour de l'axe $\underline{\boldsymbol{u}}$ (défini par le vecteur unitaire $\underline{\boldsymbol{u}}$) d'un angle $\boldsymbol{\theta}$, revient donc à tourner autour de l'axe z_j d'où la relation suivante :

$$Rot(\underline{u},\theta)T_j^i = T_j^i Rot(z,\theta) \quad (1.4)$$

Ce qui donne :

$$Rot(\underline{u},\theta) = T_j^i Rot(z,\theta){T_j^i}^{-1} \quad (1.5)$$

En remplaçant $\boldsymbol{T_j^i}$ par son expression et sachant que :

$${T_j^i}^{-1} = Rot(z,-\alpha)Rot(x,-\beta) \quad (1.6)$$

On trouve :

$$Rot(\underline{u},\theta) = Rot(z,\alpha)Rot(x,\beta)Rot(z,\theta)Rot(z,-\alpha)Rot(x,-\beta) \quad (1.7)$$

En développant cette relation on a :

$$Rot(\underline{u},\theta) = \begin{pmatrix} A(\underline{u},\theta) & 0 \\ 0 & 1 \end{pmatrix}$$

Ou :

$$A(\underline{u},\theta) = \begin{pmatrix} {u_x}^2(1-c\theta)+c\theta & u_xu_y(1-c\theta)-u_zs\theta & u_xu_z(1-c\theta)+u_ys\theta \\ u_xu_y(1-c\theta)+u_zs\theta & {u_y}^2(1-c\theta)+c\theta & u_yu_z(1-c\theta)-u_xs\theta \\ u_xu_z(1-c\theta)-u_ys\theta & u_yu_z(1-c\theta)+u_xs\theta & {u_z}^2(1-c\theta)+c\theta \end{pmatrix} \quad (1.8)$$

On préfère utiliser la relation suivante :

$$A(\underline{u},\theta) = \underline{u}\,\underline{u}^{T}(1-c\theta) + I_3c\theta + AS(\underline{u})s\theta \quad (1.9)$$

Avec :
$$AS(\underline{u}) = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix}$$

2. la représentation exponentielle

Nous avons la matrice de associée a une rotation autour d'un axe $\boldsymbol{u}$ avec un angle $\boldsymbol{\theta}$ est donnée par:

$$A(\underline{u},\theta) = \underline{u}\,\underline{u}^{T}(1-c\theta) + I_3c\theta + AS(\underline{u})s\theta$$

De plus :

$$\underline{u}\,\underline{u}^{T} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}(u_x \quad u_y \quad u_z) = \begin{pmatrix} {u_x}^2 & u_xu_y & u_xu_z \\ u_xu_y & {u_y}^2 & u_yu_z \\ u_xu_z & u_yu_z & {u_z}^2 \end{pmatrix} \quad (1.10)$$

Et :

$$AS(\underline{u})^2 = AS(\underline{u}).AS(\underline{u}) = \begin{pmatrix} -{u_y}^2-{u_z}^2 & u_x u_y & u_x u_z \\ u_x u_y & -{u_x}^2-{u_z}^2 & u_y u_z \\ u_x u_z & u_y u_z & -{u_x}^2-{u_y}^2 \end{pmatrix} \quad (1.11)$$

Et sachant que :

$$\begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \begin{pmatrix} s\alpha s\beta \\ -c\alpha s\beta \\ c\beta \end{pmatrix}$$

Il vient que : $${u_x}^2+{u_y}^2+{u_z}^2 = 1 \quad (1.12)$$

En combinant (1.11) et (1.12) on obtient :

$$AS(\underline{u})^2 = \begin{pmatrix} {u_x}^2-1 & u_x u_y & u_x u_z \\ u_x u_y & {u_y}^2-1 & u_y u_z \\ u_x u_z & u_y u_z & {u_z}^2-1 \end{pmatrix} \quad (1.13)$$

De (1.10) et (1.13) on déduit que :

$$AS(\underline{u})^2 = \underline{u}\,\underline{u}^{\mathrm{T}} - \mathrm{I}_3 \quad (1.14)$$

Ce qui donne :

$$\underline{u}\,\underline{u}^{\mathrm{T}} = AS(\underline{u})^2 + \mathrm{I}_3 \quad (1.15)$$

On remplaçant (1.15) dans (1.9) on trouve :

$$A(\underline{u},\theta) = I_3 + AS(\underline{u})\, s\theta + AS(\underline{u})^2(1-c\theta) \quad (1.16)$$

En développant en série de Mac Laurin les fonctions sinus et cosinus, on a :

$$A(\underline{u},\theta) = I_3 + AS(\underline{u})\,(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} \ldots) + AS(\underline{u})^2(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} \ldots)$$

$$A(\underline{u},\theta) = I_3 + AS(\underline{u})\,\theta + AS(\underline{u})^2\frac{\theta^2}{2!} - AS(\underline{u})\frac{\theta^3}{3!} - AS(\underline{u})^2\frac{\theta^4}{4!} + AS(\underline{u})\frac{\theta^5}{5!} \ldots \quad (1.17)$$

Et sachant que :

$$AS(\underline{u})^3 = -AS(\underline{u}) \text{ Et } AS(\underline{u})^4 = -AS(\underline{u})^2 \text{et } AS(\underline{u})^5 = AS(\underline{u}) \quad (1.18)$$

Il vient que :

$$A(\underline{u},\theta) = I_3 + AS(\underline{u})\,\theta + AS(\underline{u})^2\frac{\theta^2}{2!} + AS(\underline{u})^3\frac{\theta^3}{3!} + AS(\underline{u})^4\frac{\theta^4}{4!} + AS(\underline{u})^5\frac{\theta^5}{5!} \ldots$$

$$A(\underline{u},\theta) = I_3 + AS(\underline{u})\,\theta + \frac{(AS(\underline{u})\theta)^2}{2!} + \frac{(AS(\underline{u})\theta)^3}{3!} + \frac{(AS(\underline{u})\theta)^4}{4!} + \frac{(AS(\underline{u})\theta)^5}{5!} \ldots \quad (1.19)$$

Ce qui représente le développement en séries de Mac Laurin de la fonction exponentielle.

$$A(\underline{u},\theta) = e^{AS(\underline{u})\theta} \quad (1.20)$$

Ceci peut s'écrire comme :

$$A(\underline{u},\theta) = e^{AS(\theta\,\underline{u})} \quad (1.21)$$

Et si on pose : $\underline{\theta} = \theta\ \underline{u}$

On obtient : $A(\underline{u},\theta) = e^{AS(\underline{\theta})}$

3. quaternions et matrice de rotation:

Soit un repère $R_j$qui est le résultat de la rotation d'un repère R_i autour d'un axe quelconque **u** par un angle $\boldsymbol{\theta}$**.**

L'orientation de R_j dans le repère R_i est déterminée par quatre paramètres dits paramètres d'Euler ou quaternions qui sont :

$$w = \cos(\frac{\theta}{2})$$

$$Q_x = u_x \sin(\frac{\theta}{2})$$

$$Q_y = u_y \sin(\frac{\theta}{2})$$

$$Q_z = u_z \sin(\frac{\theta}{2})$$

Ces paramètres on les propriétés suivantes :

$$2w^2 - 1 = \cos(\theta) \tag{1.22}$$

$$w^2 + {Q_x}^2 + {Q_y}^2 + {Q_z}^2 = 1 \tag{1.23}$$

La matrice de rotation associé a ces paramètres et donnée par :

$$A(\underline{u}, \theta) = \begin{pmatrix} 2(w^2 + {Q_x}^2) - 1 & 2(Q_xQ_y - wQ_z) & 2(Q_xQ_z + wQ_y) \\ 2(Q_xQ_y + wQ_z) & 2(w^2 + {Q_y}^2) - 1 & 2(Q_yQ_z - wQ_x) \\ 2(Q_xQ_z - wQ_y) & 2(Q_yQ_z + wQ_x) & 2(w^2 + {Q_z}^2) - 1 \end{pmatrix} \tag{1.24}$$

En pose :

$$Q = \begin{pmatrix} Q_x \\ Q_y \\ Q_z \end{pmatrix}$$

On a:

$$AS(Q) = \begin{pmatrix} 0 & -Q_z & Q_y \\ Q_z & 0 & -Q_x \\ -Q_y & Q_x & 0 \end{pmatrix}$$

$$AS(Q)^2 = \begin{pmatrix} -{Q_y}^2 - {Q_z}^2 & Q_xQ_y & Q_xQ_z \\ Q_xQ_y & -{Q_x}^2 - {Q_z}^2 & Q_yQ_z \\ Q_xQ_z & Q_yQ_z & -{Q_x}^2 - {Q_y}^2 \end{pmatrix} \tag{1.25}$$

De (1.23) on a :

$$-{Q_y}^2 - {Q_z}^2 = w^2 + {Q_x}^2 - 1$$

$$-{Q_x}^2 - {Q_z}^2 = w^2 + {Q_y}^2 - 1$$

$$-{Q_x}^2 - {Q_y}^2 = w^2 + {Q_z}^2 - 1$$

Donc (1.25) devient :

$$AS(Q)^2 = \begin{pmatrix} w^2 + {Q_x}^2 - 1 & Q_xQ_y & Q_xQ_z \\ Q_xQ_y & w^2 + {Q_y}^2 - 1 & Q_yQ_z \\ Q_xQ_z & Q_yQ_z & w^2 + {Q_z}^2 - 1 \end{pmatrix}$$

Et sachant que :

$$QQ^T = \begin{pmatrix} {Q_x}^2 & 0 & 0 \\ 0 & {Q_y}^2 & 0 \\ 0 & 0 & {Q_z}^2 \end{pmatrix}$$

Il vient que :

$$A(\underline{u}, \theta) = w^2 I_3 + 2wAS(Q) + QQ^T + AS(Q)^2$$

Bibliographie

A. LEJEUNE, S. P. (2012). Utilisation de la Kinect. *Linux Magazine France* , 16-29.

Abdullah Hojaij, J. Z. (2014). A Two Phase RGB-D Visual Servoing Controller. *International Conference on Intelligent Robots and Systems.* Chicago, IL, USA: IEEE.

Céline Teuliere, E. M. (2012). Direct 3D servoing using dense depth maps. *Conf. on Intelligent Robots and Systems, IROS'12.* Vilamoura, Portugal: IEEE.

Céline Teuliere, E. M. (2010). Using multiple hypothesis in model-based tracking. *Conf. on Robotics and Automation, ICRA'10,.* Anchorage, Alaska, United.

Chaumette, F. (2002). Asservissement visuel. Dans KHALIL, *La commande des robots* . Hermès. .

hamza, d. (2015). *Conception et réalisation d'un driver pour la plateforme robotique B21r à l'aide de l'outil Ros.* algerie: USTHB.

Lapayre, B. B.-C. (2012). *Projet Kinect.* l'Université de Franche-Comté.

Martinet, P. e. (1999). *ASSERVISSEMENT VISUEL.* Universit e Blaise Pascal Clermont-Ferrand.

Martinez, A., & Farnàndez, E. (2013). *Learnining ROS For Robotics Programing.* Birmingham B3 2PB, UK: Packt Publishing.

oudah. (2012). *LE CONTRÔLE VISUEL APPLIQUÉ DANS LA ROBOTIQUE MOBILE.* ALGERIE: Thèse préparée au sein du Laboratoire de Commande des Processus de l'ENP .

Tamtsia, A. Y. (2013). *Nouvelles contributions `a l'application des moments en asservissement visuel.* france: Universite Blaise Pascal - Clermont-Ferrand II.

Wenhao Fu, H. H.-A. (2013). Visual servoing based mobile robot navigation able. *IEEE* .