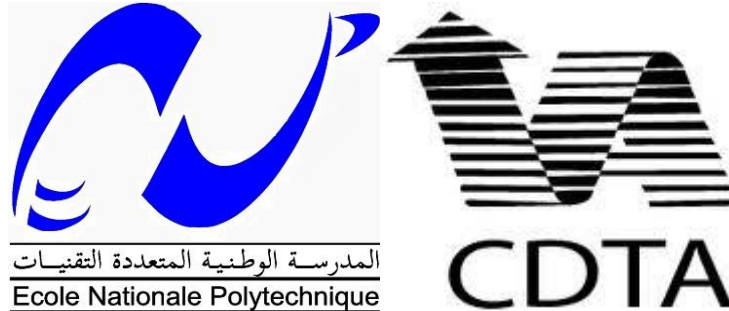


REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA
RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE



Département d'Automatique

Mémoire de Master en Automatique

Thème :

***Amélioration de la tâche du suivi implémentée au Robot
Mobile B21r***

Réalisé et présenté par:

Mahtout Imane

Proposé et dirigé par :

Pr H.Chekireb

Soutenu le 24/06/2015

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 ، قدرة 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 à la maîtrise 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 différents 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.

Sommaire

Table des Figures

Introduction	1
Chapitre 1 : Présentation et modélisation de la plateforme expérimentale B21r	3
1. Introduction.....	3
2. Présentation de la plateforme expérimentale	3
2.1 Le robot mobile B21r.....	3
2.2 Le système de vision.....	5
2.3 Le PC externe	6
3. Modélisation de la plateforme expérimentale.....	7
4. conclusion :	9
Chapitre 2 : L'asservissement visuel 3D et le calcul de la pose.....	10
1. Introduction.....	10
2. Techniques de l'asservissement visuel	10
2.1 Asservissement visuel 2D	10
2.2 Asservissement visuel 3D	11
3. La commande 3D avec retour d'état :	12
3. le calcul de la pose :	16
4. l'organigramme de la nouvelle commande :	18
5. Tests et résultats :	19
Conclusion	21
Annexe	22
Bibliographie	

Table des Figures :

Figure 1 : Robot mobile B21r	5
Figure 2 : La Kinect sans cache.....	6
Figure 3 : Organigramme de l'architecture globale du robot	7
Figure 4 : Repères du système Robot-Kinect	9
Figure 5 : Asservissement visuel 2D.....	13
Figure 6 : Représentation des repères utilisés	14
Figure 7 : Représentation des angles d'Euler	19
Figure 8 : l'organigramme de la commande.....	20
Figure 9 : vitesses du robot avec la cible en face	21
Figure 10 : vitesses du robot avec la cible de dos	22

Introduction

Les robots mobiles sont en constante évolution principalement depuis les années 2000, dans les domaines militaires (drones volants, robots sous-marins, . . .), le médical, l'agriculture, etc. Ils sont particulièrement demandés pour réaliser des tâches considérées comme pénibles, dangereuses ou impossible pour l'homme. C'est le cas par exemple des opérations de déminage, de recherche de boîtes noires d'avions de ligne au fond de l'océan ou d'exploration de planètes. Les satellites artificiels, les lanceurs, les métros sans conducteurs.

Il s'agit à terme de voir ces robots interagir de façon naturelle et effective avec les humains pour les accompagner et les guider. En effet, cette interaction ne peut se faire sans la reproduction des capacités humaines de perception et d'action en utilisant les données des capteurs extéroceptifs et plus particulièrement de celles issues d'une caméra, l'acquisition de ces données et leurs traitement permettent l'élaboration des commandes nécessaires à l'exécution des tâches robotiques d'une façon autonome .

Parmi les tâches robotiques les plus basiques pour l'interaction sociale, la tâche du suivi d'une cible mobile.

Dans le cadre de notre projet de fin d'étude s'intitulant « Conception et implémentation de programme de commande pour la plate-forme robotique B21r sous ROS » nous avons traité le problème de la tâche du suivi en utilisant les techniques d'asservissement visuel avec différentes commandes .ces techniques ont été implémentées sur la plateforme robotique B21r dotée d'un capteur RGB_D mise à notre disposition par Centre de Développement des Technologies Avancées (CDTA) .

Après différent tests avec les techniques de l'asservissement visuel 2D et 3D nous avons conclu que la commande 3D présente de meilleurs performances et assure la tâche de suivi mais son seul inconvénient est que son élaboration repose sur une approximation qui implique sa convergence dans le cas où la cible mobile est de dos ou de face dans tous le scénario si cette dernière se retourne le robot ne la suit plus. Pour remédier à ce problème la solution proposé est d'adapter cette commande à un programme de perception pouvant dire que la personne est de dos ou de face.

Le travail effectué dans le cadre de ce mémoire est présenté dans quatre (02) chapitres :

Au chapitre I, nous présentons la plateforme expérimentale B21r et sa modélisation, le capteur de vision RGB_D ainsi que les notions de géométrie projective.

Au chapitre II, nous présentons, les techniques d'asservissement visuel et l'élaboration de la loi de commande 3D, ainsi que la perception de la pose de la cible, par la suite nous présenterons les résultats des tests effectués.

Chapitre 1

Présentation et modélisation de la plateforme expérimentale B21r

1. Introduction

Afin de réaliser la tâche du suivi d'une cible mobile nous avons utilisé les techniques de l'asservissement visuel qui consiste à intégrer dans la boucle de commande des robots, des informations visuelles fournies par des caméras dans notre cas nous avons un capteur RGB_D monté sur une plateforme expérimentale B21r mis à notre disposition par le (CDTA)

Dans ce chapitre, nous présenterons la plateforme expérimentale, le capteur de vision et son principe de fonctionnement par la suite nous passons à la modélisation de la plateforme.

2. Présentation de la plateforme expérimentale

Notre système robotique comporte essentiellement 3 dispositifs à savoir : le robot mobile **B21r**, le capteur RGB_D **Kinect**, et un **PC** externe.

2.1 Le 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).



Figure 1 : Robot mobile B21r

Le B21r est doté d'un système de contrôle nommé **rFlex** 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,...).

2.2 Le système de vision

Notre système de vision est un capteur RGB_D appelé Kinect, c'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 (microphones)

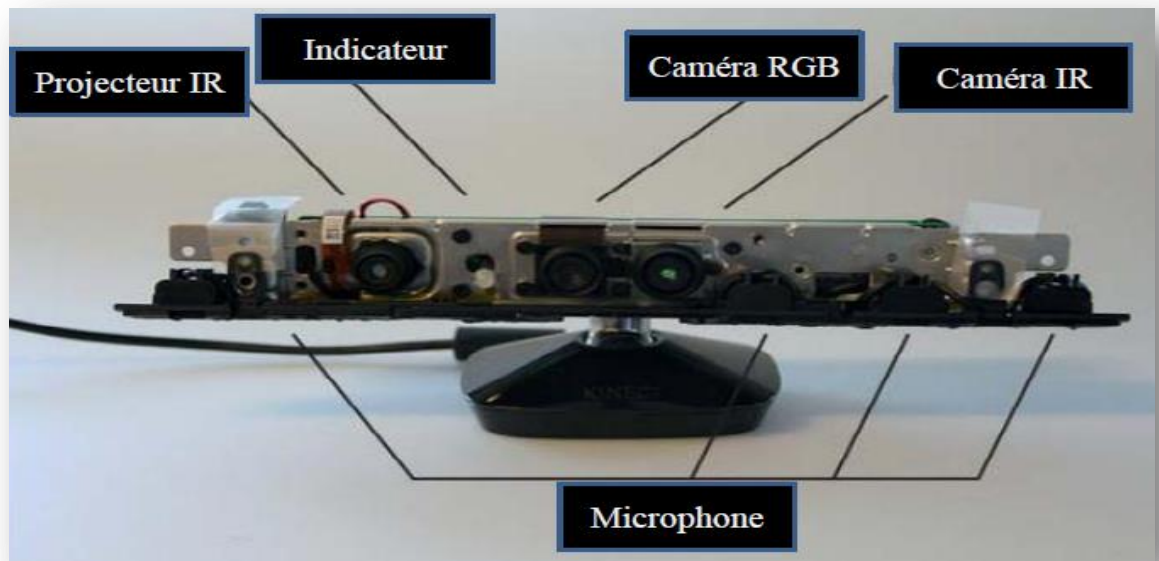


Figure 2 : La Kinect sans cache

La Kinect fournit deux images une image **RGB** qui est une image couleur où dans chaque pixel on trouve l'information sur la couleur du point correspondant dans la scène observée.

La deuxième image est une image infrarouge **IR** qui est acquise par un capteur CMOS monochrome (caméra infrarouge), équipé d'un filtre IR (infrarouge) en simultané avec la projection d'un flux laser, dans la gamme infrarouge sur la scène. 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.

Ainsi les valeurs de profondeur représentent la distance entre l'objet et le plan de la caméra IR. 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.

2.3 Le PC externe

Le B21r est d'ancienne génération où le PC embarqué, de très faibles performances, ne permet pas d'intégrer directement les capteurs externes, l'équipe Robotique du CDTA a tout simplement remplacé ce dernier par une nouvelle architecture intégrant un PC externe. Ce dernier comporte d'une part, un outil robotique très consistant **ROS** et d'autre part, les programmes assurant sa communication avec le système de commande du robot via une liaison **RS232** et il est connecté à la Kinect par une liaison **USB**.

L'organigramme suivant présente l'architecture globale de la plateforme robotique :

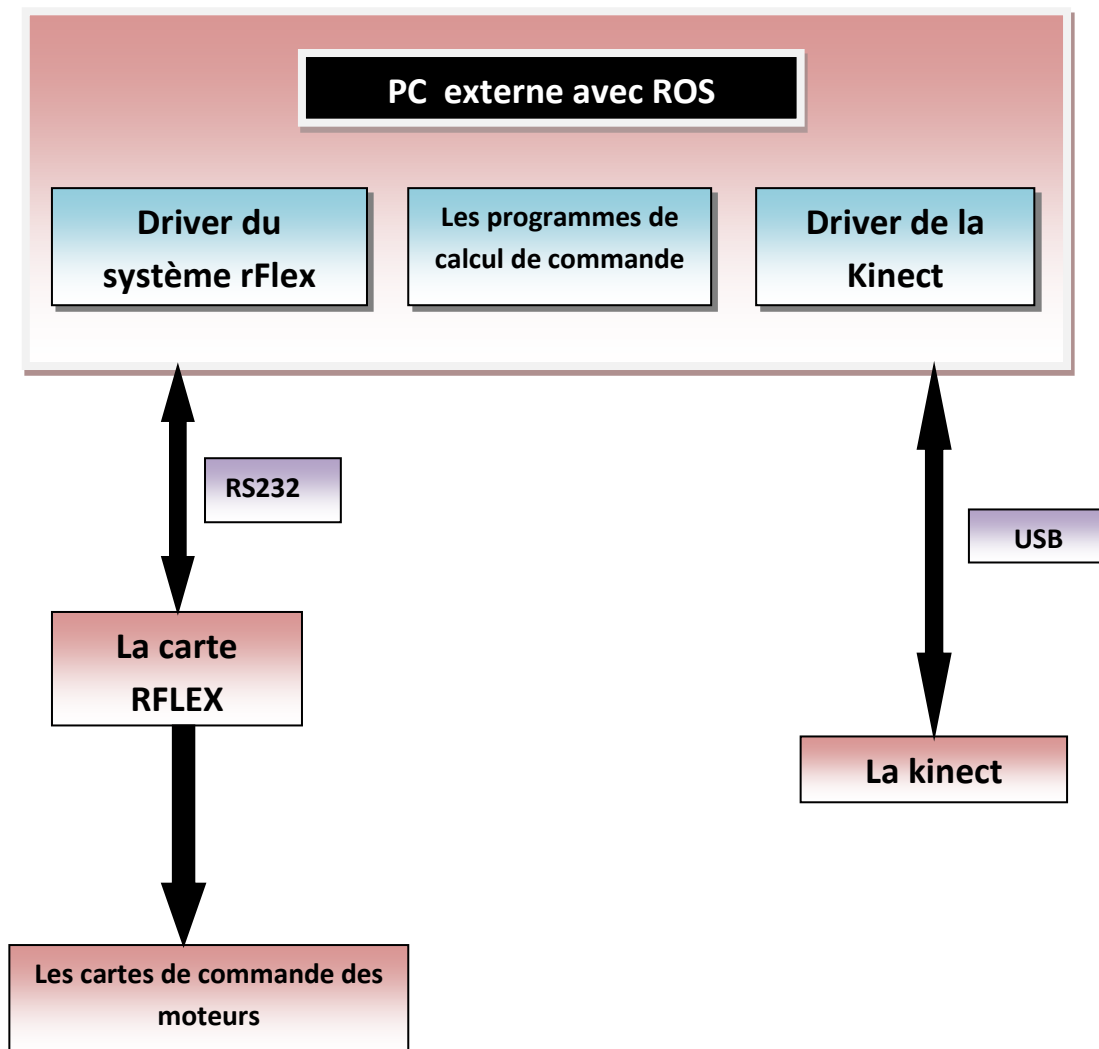


Figure 3 : Organigramme de l'architecture globale du robot

3. Modélisation de la plateforme expérimentale

Pour commander un système une phase de modélisation est nécessaire et sachant que notre système est commandable en vitesse nous avons choisis d'établir le modèle cinématique de la plateforme robotique.

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.

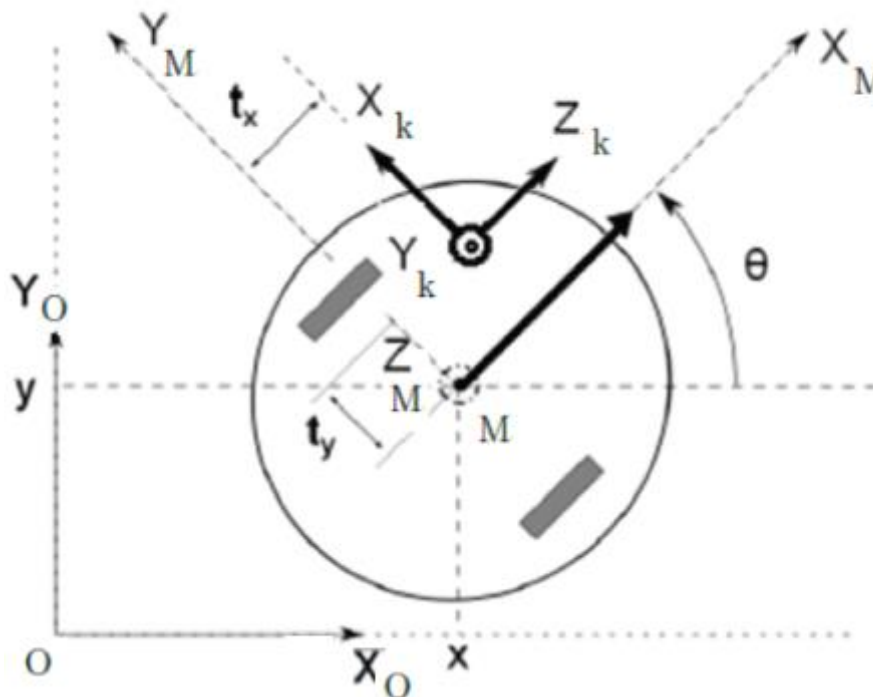


Figure 4 : Repères du système Robot-Kinect

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

$$\overrightarrow{MK} = \begin{pmatrix} t_x \\ t_y \\ 0 \end{pmatrix} \quad (1.1)$$

Le vecteur configuration du robot est défini comme suit :

$$q = \begin{pmatrix} l \\ \theta \end{pmatrix} \quad (1.2)$$

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.

On note T_{K/R_0} le torseur cinématique de la Kinect :

$$T_{K/R_0} = (V_{K/R_0}, \Omega_{K/R_0})^T \quad (1.3)$$

Avec :

$$V_{K/R_0} = (VX_K \ VY_K \ VZ_K)^T \quad (1.4)$$

$$\Omega_{K/R_0} = (\Omega X_K \ \Omega Y_K \ \Omega Z_K)^T \quad (1.5)$$

La vitesse V_{K/R_0}^{Rk} est déduite en exploitant la loi de composition des vitesses :

$$V_{K/R_0}^{Rk} = V_{K/RM}^{Rk} + V_{M/R_0}^{Rk} + \Omega_M^{Rk} \wedge \overrightarrow{RC}^{Rk} \quad (1.6)$$

Du moment que la Kinect est fixée sur le robot on a :

$$V_{K/RM}^{Rk} = \frac{d}{dt} (\overrightarrow{RC})^{Rk} = 0 \quad (1.7)$$

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 :

$$V_{M/R_0}^{Rk} = \frac{d\overrightarrow{OM}^{Rk}}{dt} \quad (1.8)$$

Donc :

$$V_{M/R_0}^{Rk} = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix} \quad (1.9)$$

On a aussi :

$$\Omega_{M/Ro}^{RC} = \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} \quad (1.10)$$

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

$$V_{K/Ro}^{Rk} = \begin{pmatrix} \dot{\theta}tx \\ 0 \\ v - \dot{\theta}ty \end{pmatrix} \quad (1.11)$$

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

$$\begin{pmatrix} V_{X_K} \\ V_{Y_K} \\ V_{Z_K} \\ \Omega_{X_K} \\ \Omega_{Y_K} \\ \Omega_{Z_K} \end{pmatrix} = \begin{pmatrix} 0 & tx \\ 0 & 0 \\ 1 & -ty \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v \\ \dot{\theta} \end{pmatrix} \quad (1.12)$$

En éliminant les mouvements inadmissibles on obtient le jacobien réduit du système :

$$J_r = \begin{pmatrix} 0 & tx \\ 1 & -ty \\ 0 & 1 \end{pmatrix} \quad (1.13)$$

4. conclusion :

Dans ce chapitre nous avons présenté l'architecture globale de notre plateforme robotique qui comporte trois parties : une unité de perception qui utilise un capteur de vision Kinect permettant l'acquisition des coordonnées 3D de la cible, d'une unité de traitement constituée par un PC externe et d'un actionneur qui est le robot mobile B21r.

Dans le prochain chapitre nous présenterons la perception de la cible et le calcul de sa pose ainsi que la loi de commande de l'asservissement visuel 3D adapté à la pose de la cible.

Chapitre 2

L'asservissement visuel 3D et le calcul de la pose

1. Introduction

La tâche du suivi d'une cible mobile nécessite des commandes qui rentrent dans le cadre de la commande par vision, cette dernière repose sur les techniques d'asservissement visuel qui se différencient par le type des informations visuelles traitées à savoir les informations 2D et 3D.

Dans ce chapitre nous présentons les différentes techniques et nous détaillerons la commande 3D par la suite nous montrerons la nécessité d'un calcul de la pose de la cible pour assurer la tâche du suivi et initier à l'interaction social entre le robot et la cible.

2. 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 maîtriser le mouvement d'un robot et ceci dans le but de réaliser différentes tâches 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.

2.1 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.

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 courantes vers la position désirées*. 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 s vers leurs valeurs désirées s^* . On choisit généralement N primitives de type point avec $N \geq 4$

Primitives désirées S^*

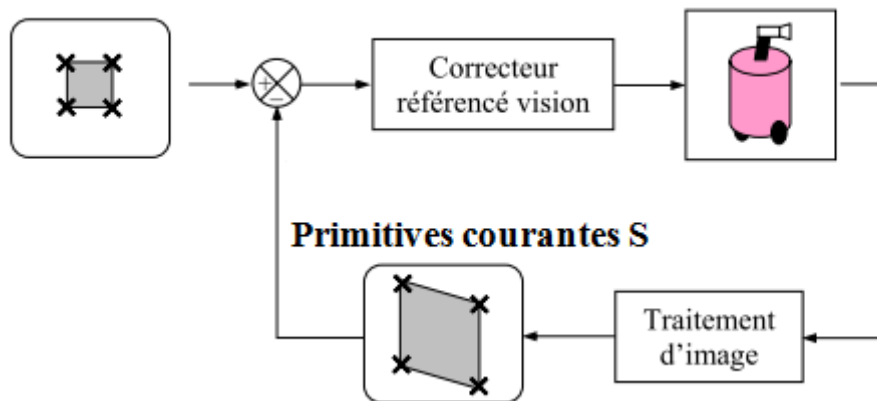


Figure 1 : Asservissement visuel 2D

2.2 Asservissement visuel 3D

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

Les commandes 3D 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.

3. La commande 3D avec retour d'état :

Nous avons synthétisé 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 avons choisi de ramener le repère Kinect courant dans une situation désirée dite repère Kinect désirée. Cette tâche 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

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

- $M_o(t)$ matrice de transformation homogène entre le repère Kinect désiré et la cible ;
- $M_p(t)$ matrice de transformation homogène entre le repère Kinect courant et le repère de la cible ;
- $M_c(t)$ matrice de transformation homogène entre le repère Kinect désiré et le repère Kinect courant.

De la Figure 6, on voit qu'il est possible de déduire la matrice $M_c(t)$ de la manière suivante :

$$M_c(t) = M_o M_p(t)^{-1}$$

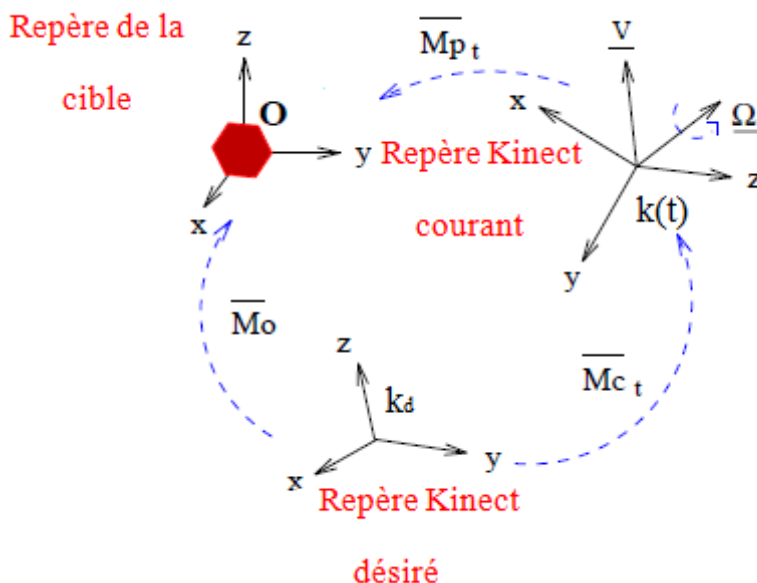


Figure 2 : Représentation des repères utilisés

Le deux matrices M_o et $M_p(t)$ sont calculées à partir des données issues d'un programme de perception qui donne les coordonnées de l'origine du repère lié au torse de la cible dans le repère de la Kinect et les quaternions associé à ce dernier.

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

$$\overline{M}_c(t) = \begin{pmatrix} R(t) & k_d k(t)_{k_d} \\ 0 & 1 \end{pmatrix} \quad (2.1)$$

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 exponentielle de la matrice d'orientation (Annexe A) :

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

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

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

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 (2.1), ce qui conduit à :

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

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 ;

$$As\left(\underline{y}(t)\right) = \frac{1}{2} * (R^T(t) - R(t)) = \sin(\|\underline{\vartheta}\|). As(\underline{u}(t)) \quad (2.2)$$

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.

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

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

Où $\overline{v}$ est le tenseur antisymétrique associé au vecteur $v = (Qx, Qy, Qz)$ 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:

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

Avec :

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

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

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

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

$$\frac{d}{dt}\underline{X}(t) = B(\underline{X}).\underline{U} \quad (2.3)$$

Ou :

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

$$\underline{U}(t) = (\underline{V}^T, \underline{\Omega}^T)^T$$

Pour assurer la convergence, on impose au système (2.3), la dynamique linéaire suivante :

$$\frac{d}{dt}\underline{X}(t) = -K\underline{X} \quad (2.4)$$

Tenant compte de (2.3) et (2.4), on en déduit que :

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

Et donc on a :

$$\underline{U} = -B(\underline{X})^+ \bar{K}\underline{X}$$

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

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

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

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

$$\underline{U} = \begin{pmatrix} V_{X_K} \\ V_{Z_K} \\ \Omega_{Y_K} \end{pmatrix}$$

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 :

$$\begin{pmatrix} V_{X_K} \\ V_{Z_K} \\ \Omega_{Y_K} \end{pmatrix} = J_r \begin{pmatrix} v \\ \dot{\vartheta} \end{pmatrix}$$

Donc :

$$\begin{pmatrix} v \\ \dot{\vartheta} \end{pmatrix} = J_r^+ \begin{pmatrix} V_{X_K} \\ V_{Z_K} \\ \Omega_{Y_K} \end{pmatrix}$$

$$\dot{q} = J_r^+ \underline{U}$$

Dans le cadre de notre projet de fin d'étude cité dans l'introduction générale nous avons testé cette loi de commande ainsi que d'autres techniques d'asservissement visuel 2D et 3D et ces derniers ont montré la convergence de cette commande et qu'elle assure la tâche du suivi mais son inconvénient majeur est que le modèle d'état mis en jeu est établie sous la condition que $\|\underline{\vartheta}\| < \frac{\pi}{2}$ ce qui veut dire cette commande est fiable dans le cas où la cible reste ou de dos ou de face si elle tourne d'un angle supérieur à $\frac{\pi}{2}$ par rapport à la consigne le robot ne suivra plus la cible et il tourne avec une vitesse constante. Pour remédier à ce problème la solution proposée est de faire un calcul de pose de la cible si la cible est de dos le robot la suit sinon il s'arrête.

3. le calcul de la pose :

Le calcul de la pose de la cible repose sur le calcul des angles d'Euler et sachant que les données sur l'orientation de la cible sont issues de la Kinect sous forme de quaternions nous avons besoin de la relation de passage quaternions /angles d'Euler.

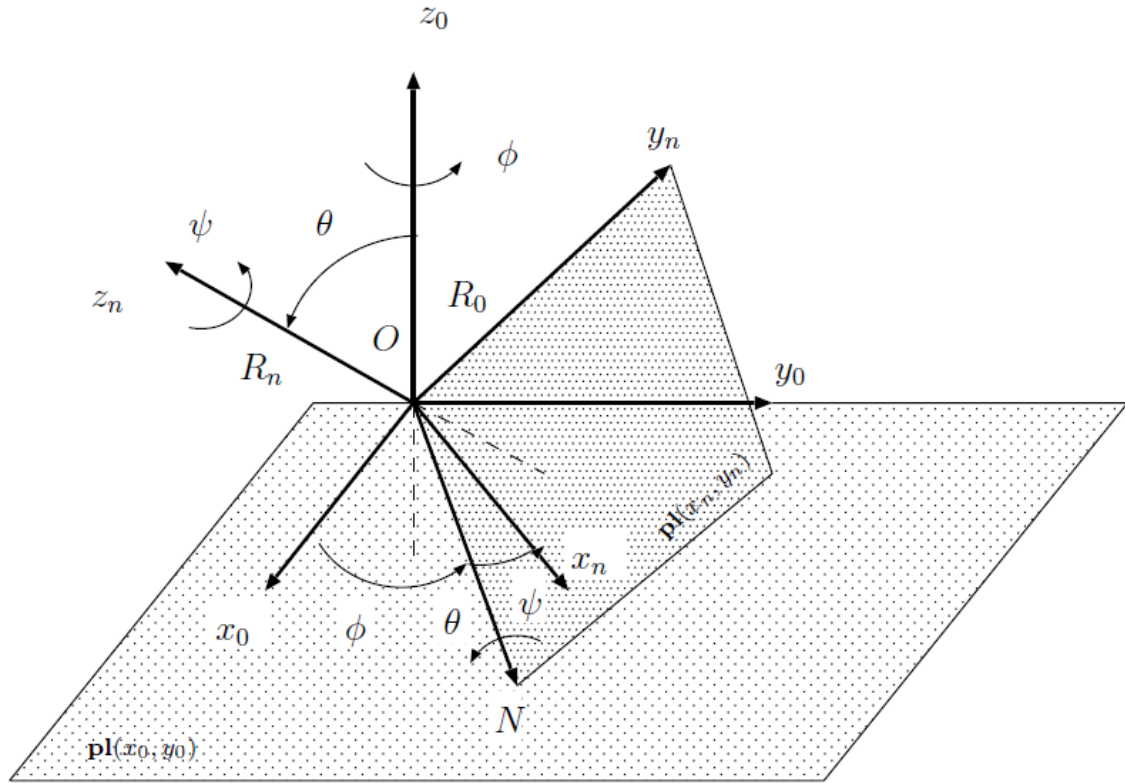


Figure 3 : Représentation des angles d'Euler

Comme le montre la figure 7, les angles d'Euler sont définis comme suit :

- φ (précession) : angle entre x_0 et ON tel que $0 \leq \varphi \leq 360^0$
- θ (nutation) : angle entre z_0 et z_n tel que $0 \leq \theta \leq 180^0$
- ψ (rotation propre) : angle entre ON et x_n tel que $0 \leq \psi \leq 360^0$

φ et ψ sont mesurés dans le sens donné par la règle du “tire-bouchon” respectivement autour des axes Z_0 et Z_n .

La composition de ces trois rotations permet de calculer la matrice d'orientation.

En effet, on a :

$$R_{\text{Eul}} = E(z, \varphi).A(x, \theta).A(z, \psi)$$

d'où

$$R_{\text{Eul}} = \begin{pmatrix} c\varphi.c\psi - s\varphi.c\theta.s\psi & -c\varphi.s\psi - s\varphi.c\theta.c\psi & s\varphi.s\theta \\ s\varphi.c\psi + c\varphi.c\theta.s\psi & -s\varphi.s\psi + c\varphi.c\theta.c\psi & -c\varphi.s\theta \\ s\theta.s\psi & s\theta.c\psi & c\theta \end{pmatrix} \quad (2.5)$$

Et sachant que la même matrice de rotation est exprimée par les quaternions comme suit (voir Annexe B)

$$R = \begin{pmatrix} 2(w^2 + Q_x^2) - 1 & 2(Q_x Q_y - w Q_z) & 2(Q_x Q_z + w Q_y) \\ 2(Q_x Q_y + w Q_z) & 2(w^2 + Q_y^2) - 1 & 2(Q_y Q_z - w Q_x) \\ 2(Q_x Q_z - w Q_y) & 2(Q_y Q_z + w Q_x) & 2(w^2 + Q_z^2) - 1 \end{pmatrix} \quad (2.6)$$

En identifiant (2.5) et (2.6) on obtient :

$$\begin{aligned} \varphi &= \arctan\left(\frac{Q_x Q_z + w Q_y}{-Q_y Q_z + w Q_x}\right) \\ \theta &= \arccos(2(w^2 + Q_z^2) - 1) \\ \psi &= \arctan\left(\frac{Q_x Q_z - w Q_y}{Q_y Q_z - w Q_x}\right) \end{aligned}$$

Après calcul des angles d'Euler et on respectant le repère de la Kinect on arrive à déduire la pose de la cible suivant le tableau suivant :

et sachant que la pose de la personne dépend de son orientation dans le plan nous considérons l'angle φ .

L'angle φ (deg)	$\varphi < -30^0$ et $\varphi > 30^0$	$30^0 < \varphi < 150^0$	$-150^0 < \varphi < -30^0$	$\varphi < -150^0$ et $\varphi > 150^0$
La pose	face	droite	gauche	dos

4. l'organigramme de la nouvelle commande :

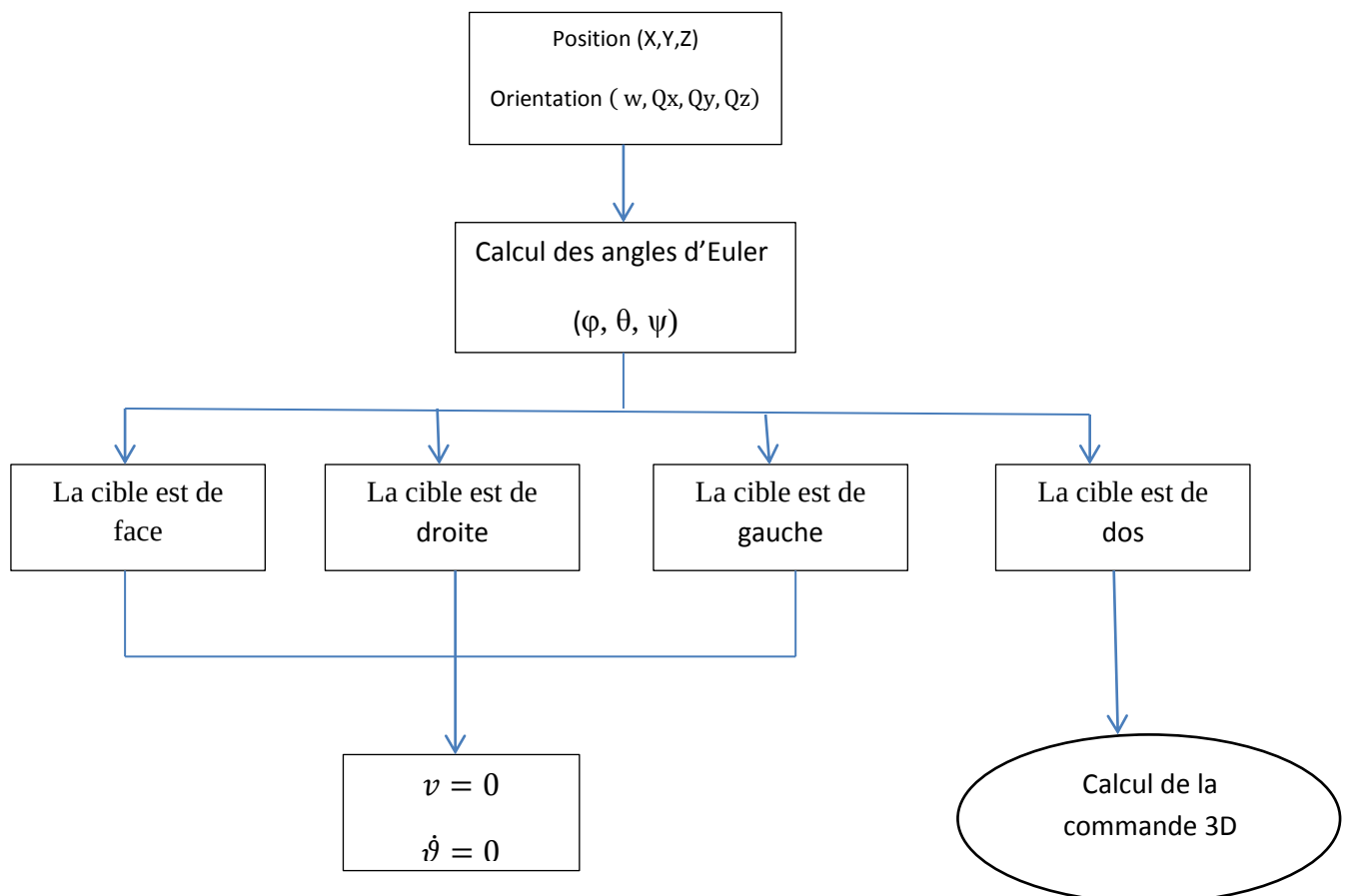


Figure 4 : l'organigramme de la commande

5. Tests et résultats :

Après l'implémentation de la tâche du suivi en respectant l'organigramme précédent le robot a pu suivre la cible en étant de dos et si elle tourne vers lui il s'arrête, les deux images suivantes ont été prises lors des tests.

The image shows two terminal windows from a robot named 'imen' on an 'Aspire-E1-571' system. The left window displays a log of events including user management and calibration attempts. The right window shows real-time position and velocity data for the robot's end effector.

```

[INFO] [1434899985.350785958]: Lost user 2
[INFO] [1434899985.390896135]: New User 2
[INFO] [1434899985.391410733]: Calibration started for user 2
[INFO] [1434899985.419464205]: Lost user 2
[INFO] [1434899985.922467251]: New User 2
[INFO] [1434899985.922675805]: Calibration started for user 2
[INFO] [1434899986.718848772]: Lost user 2
[INFO] [1434899988.258225119]: New User 2
[INFO] [1434899988.258509206]: Calibration started for user 2
[INFO] [1434899990.760854183]: Lost user 2
[INFO] [1434899991.429571457]: New User 2
[INFO] [1434899991.429793192]: Calibration started for user 2
[INFO] [1434899996.432653362]: Calibration failed for user 2
[INFO] [1434899996.432859525]: Calibration started for user 2

-0.166111 -0.131431 0
0.364302 0.912797 0
0 0 -0.0830556
0 0 -0.914577
0 0 -0.0657156
0.903395 -0.415895 0
-0.171944 -0.127964 0
0.392853 0.900376 0
0 0 -0.0859721
0 0 -0.901885
0 0 -0.0639819
0.896359 -0.428441 0
-0.181484 -0.119925 0
0.404507 0.895588 0
0 0 -0.090742
0 0 -0.895974
0 0 -0.0599626
-0.43419 -0.887266 0
-0.314422 0.311228 0
0.844168 -0.340437 0
0 0 -0.157211
0 0 0.387313
0 0 0.155614
  
```

```

y = -0.0633687
z = 1.61383
lost_usrs:1*****
*****
The Position : Face !!
z = 3.70594
x = 0.2158
y = -0.0646203
z = 1.61108
lost_usrs:1*****
*****
The Position : Face !!
z = 3.7286
x = 0.215589
y = -0.0671182
z = 1.60657
lost_usrs:1*****
*****
The Position : Face !!
z = 3.76125
x = 0.21145
y = -0.0659912
z = 1.59725

angular:
x: 0.0
y: 0.0
z: 0.0
--
linear:
x: 0.0
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: 0.0
--
  
```

Figure 5 : vitesses du robot avec la cible en face

```

[INFO] [1434899988.258509206]: Calibration started for user 2
[INFO] [1434899990.760854183]: Lost user 2
[INFO] [1434899991.429571457]: New User 2
[INFO] [1434899991.429793192]: Calibration started for user 2
[INFO] [1434899996.432653362]: Calibration failed for user 2
[INFO] [1434900001.468004624]: Calibration started for user 2
[INFO] [1434900001.468190036]: Calibration failed for user 2
[INFO] [1434900006.506462686]: Calibration started for user 2
[INFO] [1434900006.506669863]: Calibration failed for user 2
[INFO] [1434900006.708121643]: Lost user 2
[INFO] [1434900010.445081943]: Lost user 1
[INFO] [1434900011.082497285]: New User 1
[INFO] [1434900011.082811394]: Calibration started for user 1

0.059038  -0.0183739  0
0.0800379  0.996635  0
0 0 0.029519
0 0 -0.995807
0 0 -0.00918695
0.995468  -0.0707086  0
0.0619376  -0.0177011  0
0.0716776  0.997305  0
0 0 0.0309688
0 0 -0.996386
0 0 -0.00885057
0.995581  -0.0669756  0
0.0649432  -0.0126364  0
0.0676584  0.997663  0
0 0 0.0324716
0 0 -0.996622
0 0 -0.00631822
0.995486  -0.0661877  0
0.0670673  -0.0162754  0
0.0671367  0.997673  0
0 0 0.0335336
0 0 -0.996579
0 0 -0.00813768

y = -0.207261
z = 1.929
lost_usrs:0*****
*****
The Position  Back !!
z = 174.749
x = 0.215638
y = -0.213883
z = 1.93916
lost_usrs:0*****
*****
The Position : Back !!
z = 174.561
x = 0.212266

angular:
x: 0.0
y: 0.0
z: -0.019944306091
---
linear:
x: 0.202585558537
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: -0.0182569624509
---
linear:
x: 0.207032034275
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: -0.0180628385789
---

```

Figure 6 : vitesses du robot avec la cible de dos

Dans ces deux images on remarque que si la cible est de face les vitesses sont nulles et si elle est de dos le robot la suit avec une vitesse calculée à partir de la commande 3D.

Conclusion

Dans le cadre de ce mémoire, nous avons amélioré la tâche du suivi d'une cible mobile par 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 2D et 3D.

Par la suite nous avons expliqué le développement d'une commande 3D qui a pour but de faire converger le repère de la Kinect vers un repère désiré déduit à partir de la pose désirée de la cible le calcul de cette commande est fondé sur une représentation d'état calculée à partir des transformations homogènes entre les repères liés à la Kinect et la cible en utilisant des approximation qui rendent le modèle d'état valable juste au voisinage de l'orientation désiré.

Pour pallier à ce problème la solution proposée est d'établir un calcul de la pose de la cible et si celle-ci n'est pas de dos on impose des vitesses nulles pour que la cible puisse interagir avec le robot.

L'implémentation de cette solution a réussi à rendre la tâche du suivi plus naturelle et le robot pourra interagir avec la cible si elle est en face de lui sinon il la suit si elle est de dos.

Annexe

Eléments de la robotique

1. Matrice de rotation autour d'un axe $\underline{u}$ quelconque :

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

Soit R_j , un repère tel que :

- son origine coïncide avec celle du vecteur $\underline{u}$,
- son axe $\underline{Z}_j$ se confonde avec le vecteur $\underline{u}$.

La transformation homogène permettant le passage du repère R_i au repère R_j peut être décomposée en :

- $Rot(\underline{z}, \alpha)$: une rotation autour de l'axe $\underline{z}$ d'un angle α tel que $\underline{u} \in$ au plan $(\underline{z}_f, \underline{y}_f)$,
- $Rot(\underline{x}, \beta)$: une rotation autour de l'axe $\underline{x}$ d'un angle β .

On obtient la relation suivante :

$$T_j^i = Rot(\underline{z}, \alpha) \cdot Rot(\underline{x}, \beta) \quad (1.1)$$

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

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

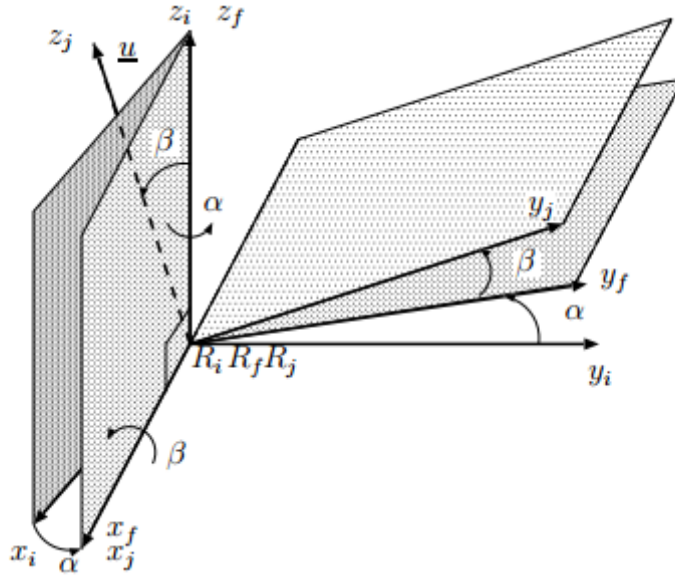


Fig 1 : rotation du repère R_i autour d'un axe $\underline{u}$

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

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

A partir de la troisième colonne de la matrice T_j^i , on peut donc extraire les coordonnées du vecteur $\underline{u}$, exprimées dans le repère 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 $\underline{u}$ (défini par le vecteur unitaire $\underline{u}$) d'un angle θ , revient donc à tourner autour de l'axe z_j d'où la relation suivante :

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

Ce qui donne :

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

En remplaçant T_j^i par son expression et sachant que :

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

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 :

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

Où :

$$A(\underline{u}, \theta) = \begin{pmatrix} u_x^2(1 - c\theta) + c\theta & u_x u_y(1 - c\theta) - u_z s\theta & u_x u_z(1 - c\theta) + u_y s\theta \\ u_x u_y(1 - c\theta) + u_z s\theta & u_y^2(1 - c\theta) + c\theta & u_y u_z(1 - c\theta) - u_x s\theta \\ u_x u_z(1 - c\theta) - u_y s\theta & u_y u_z(1 - c\theta) + u_x s\theta & u_z^2(1 - c\theta) + c\theta \end{pmatrix} \quad (1.8)$$

On préfère utiliser la relation suivante :

$$A(\underline{u}, \theta) = \underline{u} \underline{u}^T (1 - c\theta) + I_3 c\theta + AS(\underline{u}) s\theta \quad (1.9)$$

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 $\mathbf{u}$ avec un angle θ est donnée par:

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

De plus :

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

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 :

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

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

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 :

$$AS(\underline{u})^2 = \underline{u} \underline{u}^T - I_3 \quad (1.14)$$

Ce qui donne :

$$\underline{u} \underline{u}^T = AS(\underline{u})^2 + I_3 \quad (1.15)$$

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 :

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

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

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 :

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

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

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 $\underline{u}$ par un angle θ .

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

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

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

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

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

Ces paramètres ont les propriétés suivantes :

$$2w^2 - 1 = \cos(\theta) \quad (1.22)$$

$$w^2 + Q_x^2 + Q_y^2 + Q_z^2 = 1 \quad (1.23)$$

La matrice de rotation associée à ces paramètres est donnée par :

$$A(\underline{u}, \theta) = \begin{pmatrix} 2(w^2 + Q_x^2) - 1 & 2(Q_x Q_y - w Q_z) & 2(Q_x Q_z + w Q_y) \\ 2(Q_x Q_y + w Q_z) & 2(w^2 + Q_y^2) - 1 & 2(Q_y Q_z - w Q_x) \\ 2(Q_x Q_z - w Q_y) & 2(Q_y Q_z + w Q_x) & 2(w^2 + Q_z^2) - 1 \end{pmatrix} \quad (1.24)$$

En pose :

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

On a :

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

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

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 :

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

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* .