INTERACTION ET OPTIMISATION DE TEMPS AVEC DES UR5 ET MIR 100

Julien Bénétier, Fabrice Duval

CONFERE'17
6 ET 7 JUILLET 2017, SEVILLE, ESPAGNE.

INTERACTION ET OPTIMISATION DE TEMPS AVEC DES UR5 ET MIR 100

Julien Bénétier[1], Fabrice Duval[2]
[1] FIA.Cesi, julien.benetier@viacesi.fr, jbenetier@cesi.fr
[2] Lineact Laboratory at Cesi, fduval@cesi.fr

Mots clés :Système d'exploitation, Robotique, Interface,véhicule à guidage autonome, Bras robot,

1. INTRODUCTION

Le terme robot vient du mot tchèque robota, généralement traduit par « travail forcé ». Cela décrit la majorité des robots. La plupart des robots dans le monde sont conçus pour les travaux de fabrication lourds et répétitifs [1]. Ils s'occupent de tâches difficiles, dangereuses ou ennuyeuses pour les êtres humains.

Le robot de fabrication le plus courant est le bras robotique. Un bras robotique typique est composé de sept segments de métal, joints par six joints. L'ordinateur commande le robot en faisant tourner des moteurs électriques reliés à chaque articulation (certains plus gros bras utilisent l'hydraulique ou pneumatique.

Ce papier propose un modèle permettant de faciliter la réalisation d'un message via un Socket, à optimiser le temps de déplacement et d'efficacité avec des robots industriels

2. ETAT DE L'ART

Il est important avant de proposer un modèle d'identifier les différents facteurs qui peuvent être évalués lors de la réalisation d'une interaction avec des UR5 et MIR 100.

Selon [2], "apprentissage de la trajectoire" est un élément fondamental d'un système de programmation par un système de démonstration, où souvent le but même de la démonstration est d'enseigner des modèles de manipulation complexes. Cependant, les manifestations humaines sont inévitablement bruyantes et incohérentes. Cet article souligne la composante d'apprentissage de la trajectoire d'un système pour des tâches de manipulation englobant la possibilité de regrouper, sélectionner et approcher les trajectoires humaines démontrées. La technique proposée offre certains avantages par rapport à d'autres approches et convient à l'apprentissage à la fois par des démonstrations individuelles et multiples.

Un autre article [3], présente une approche systématique pour représenter et estimer les erreurs de positionnement cartésiennes de robots manipulateurs avec des fonctions analytiques telles que les polynômes de Fourier et les polynômes ordinaires. Un manipulateur robot Motoman SK 120 a été utilisé comme système expérimental pour évaluer l'efficacité de l'approche. Comme partie complémentaire de ce processus d'évaluation, les paramètres cinématiques du système expérimental sont également identifiés.

Robot Operating System (ROS) [4], est un ensemble d'outils informatiques open source permettant de développer des logiciels pour la robotique. À l'origine, il est développé en 2007 par la société

américaine Willow Garage, pour son robot PR2 (Personal Robot 2). Son développement est aujourd'hui mené par l'Open Source Robotics Foundation (OSRF). ROS est officiellement supporté par plus de 75 robots.

Selon [5], cet article vise à présenter une nouvelle approche de programmation de robots hors ligne pour la génération automatisée des trajectoires sur des surfaces libres destinées à l'application de peinture par pulvérisation .la trajectoire est générée sur des surfaces réelles où la vitesse optimale est de calculer à l'aide d'un algorithme génétique en considérant des paramètres tels que le modèle de surface, le modèle de pulvérisateur, le modèle de distribution de peinture et des contraintes de taches.

L'ensemble de ces articles permettent ainsi de proposer un modèle complet d'interaction simplifiant au maximum le développement de l'apprentissage d'une trajectoire.

3. MODELE

Dans le cadre d'un travail manuel, notre bras va déplacer un objet d'un endroit à l'autre. De même, le travail du bras robotique consiste à déplacer un effecteur d'extrémité d'un endroit à un autre. Nous pouvons habiller les bras robotiques avec toutes sortes d'effecteurs finaux, qui sont adaptés à une application particulière. Un effecteur d'extrémité commun est une version simplifiée de la main, qui peut saisir et porter des objets différents. Les mains robotisées ont souvent des capteurs de pression intégrés qui indiquent à l'ordinateur à quel point le robot saisit un objet particulier. Cela empêche le robot de tomber ou de casser ce qu'il porte. D'autres effecteurs d'extrémité incluent les chalumeaux, les perceuses et les peintres de pulvérisation.

Les robots industriels sont conçus pour faire exactement la même chose, dans un environnement contrôlé, encore et encore. Pour apprendre à un robot comment faire son travail, le programmeur guide le bras à travers les mouvements en utilisant un contrôleur de poche. Le robot stocke la séquence exacte des mouvements dans sa mémoire, et le fait à chaque fois qu'une nouvelle unité descend la ligne d'assemblage.

1) Système d'exploitation

Pour manipuler les robots nous avons à notre disposition plusieurs interfaces qui sont ROS et Socket Test.

Le système d'exploitation ROS [4] est une plateforme de développement logicielle pour robot. Il s'agit d'un méta-système d'exploitation qui peut fonctionner sur un ou plusieurs ordinateurs et qui fournit plusieurs fonctionnalités : abstraction du matériel, contrôle des périphériques de bas niveau, mise en œuvre de fonctionnalités couramment utilisées, transmission de messages entre les processus et gestions des packages installés.

ROS offre une architecture souple de communication inter-processus et inter-machine. Les processus ROS sont appelés des *nodes* et chaque *node* peut communiquer avec d'autres *via* des *topics*. La connexion entre les *nodes* est gérée par un *master* et suit le processus suivant :

1=> Un premier *node* avertit le *master* qu'il a une donnée à partager

2=> Un deuxième *node* avertit le *master* qu'il souhaite avoir accès à une donnée

3=> Une connexion entre les deux *nodes* est créée

4=> Le premier *node* peut envoyer des données au second

Le système SocketTest [6] est une interface logicielle avec les services du système d'exploitation, grâce à laquelle un développeur exploitera facilement et de manière uniforme les services d'un protocole réseau.

2) L'interface contrôleur

Universal Robots 5 (UR5) : Les interfaces des robots permettent de spécifier les commandes des contrôleurs pour activer les différentes actions. Il peut s'agir de commandes simples (juste un clic) ou bien des commandes plus complexes (mouvements précis attendus).

MIR 100 : Les interfaces des véhicules à guidage autonome servent à manipuler les véhicules, cartographier les pièces, programmer des trajectoires et déplacer le véhicule manuellement via un joystick…

Universal Robot 5 (UR5) sur le MIR 100 : On n'a pas d'accès à l'interface homme machine de l'Universal Robot 5 puisqu'elle est supprimée. De plus, lorsque nous nous connectons au MIR 100 par l'interface Wifi global du laboratoire le système bloque l'accès à l'interface web de l'UR5. A ce jour, nous n'avons pas trouvé la solution pour manipuler les deux systèmes en même temps via ROS. Nous choisissons d'utiliser des commandes directes via un port Socket ouvert sur l'UR 5 en utilisant l'interface logicielle SocketTest.

3) Action

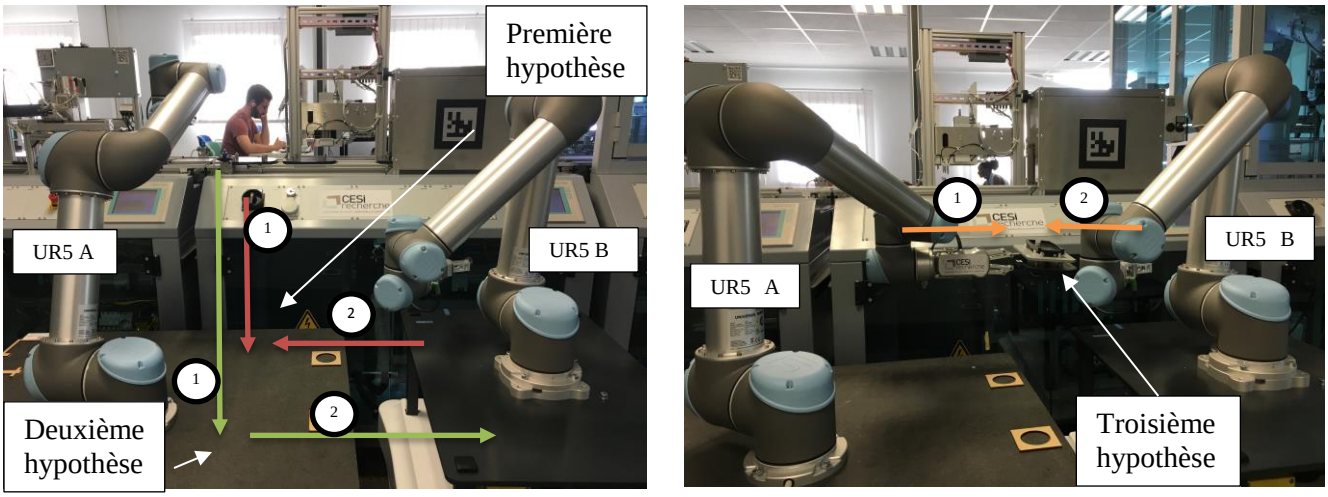


Figure 1: Photos du robot UR5 et MIR 100 avec adaptation UR5

Le modèle choisi permet d'estimer quelles méthodes est la mieux appropriée pour déplacer un objet d'une place à une autre en utilisant un bras robot UR5 et un véhicule à guidage autonome MIR 100 voir (Figure 1).

- La première hypothèse : Avec l'UR5 A on prendre la pièce sur la chaine de production ensuite l'UR5 A le dépose sur sa table et récupérer par l'UR5 B du MIR 100 pour l'emmener la pièce au poste de maintenance.

- La seconde hypothèse : Avec l'UR5 A on prendre la pièce sur la chaine de production que l'on vient déposer directement sur la table de l'UR5 B du MIR100.

- La troisième hypothèse : Avec l'UR5 A on prendre la pièce sur la chaine de production dans la suite du mouvement du bras robot, le préhenseur de l'UR5 B du mir 100 viens prendre immédiatement l'objet dans la pince du bras robot A pour l'emmener après à son poste de maintenance.

Le positionnement du MIR 100 n'est pas toujours précis et nous avons besoin de précision pour récupérer l'objet avec les deux préhenseurs. J'ai donc choisi de faire la seconde hypothèse puisque les deux autres sont plus difficiles à accomplir.

4. EXPERIMENTATION

1) ROS avec l'UR5

Des démonstrations on était faites avec le système d'exploitation ROS sur un bras robot. Plus tard tous nos robots fonctionneront sûr qu'une interface qui est le ROS. Sa permettra d'être plus libre sur la programmation pour exploiter nos robots. En installant des package sur ROS ça permet d'avoir aussi une interface du bras. Le ROS va permette d'avoir un meilleur déplacement du bras étant donné que l'interface de UR5 n'est pas toujours intuitive ou fermé et permet d'avoir un panel d'algorithme pour profiter au maximum de ces capacités.

Pour faire fonctionner le bras robot sur ROS nous avons besoin que la machine et le Pc soit sur le même réseau et d'installer des package. Le premier est le package « Universal Robots » qui est l'interface du robot dans le ROS. Ensuite il contient plusieurs petits package comme « UR Bringup » qui permet d'importer le robot réel dans la machine. Après il y a aussi le package « UR Moveit config » c'est lui qui va permettre d'envoyé les commande au robot. Le package « UR Description » comporte tous les fichiers descriptifs. Grace à tous ces package on peut faire des simulations en temps réel.

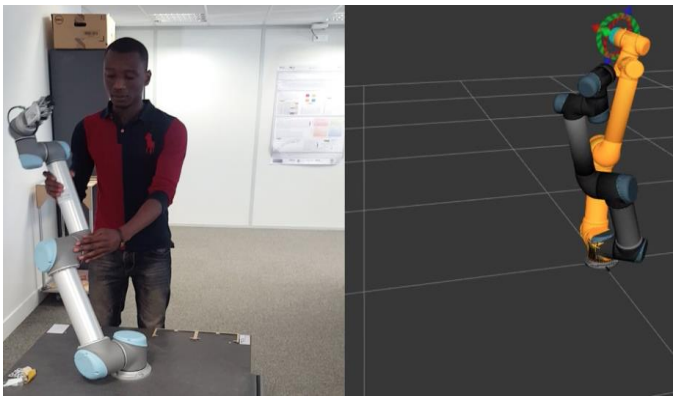


Figure 2: Interface du bras robot avec ROS

2) SocketTest avec l'UR5

Le SocketTest est commandé en directe via un câble Ethernet en configurant l'adresse IP sur l'interface de l'Universal Robot et ensuite de tester la communication du SocketTest. Aujourd'hui j'ai utilisé SocketTest qu'en envoyant des points de déplacement.

La notion de socket a été introduite dans les distributions de Berkeley (un fameux système de type UNIX, dont beaucoup de distributions actuelles utilisent des morceaux de code), c'est la raison pour laquelle on parle parfois de sockets BSD (Berkeley Software Distribution).

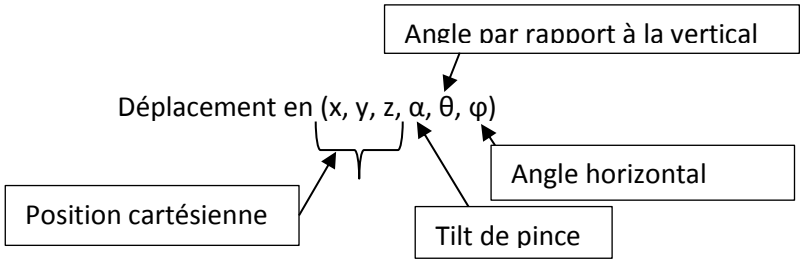
Il s'agit d'un modèle permettant la communication inter processus (IPC - Inter Process Communication) afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP.

Pour la configuration du socket « serveur » via à l'UR 5, il n'y en a pas puise qu'il est régler directement par défaut car il ouvre un port socket automatiquement.

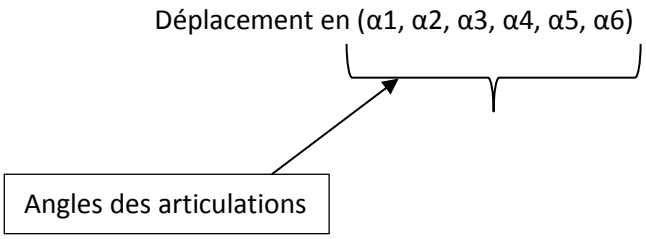
Configuration du socket « client » => android => SocketTest => configuration du port ouvert, comme d'un port d'accès. Le mode connecté (comparable à une communication téléphonique), utilisant le protocole TCP. Dans ce mode de communication, une connexion durable est établie entre les deux processus, de telle façon que l'adresse de destination n'est pas nécessaire à chaque envoi de données. Les sockets sont implémentés dans différents langages (C, Java, ...). En langage C, elles utilisent des fonctions et des structures disponibles dans la bibliothèque <sys/socket.h>

Envoie des commandes :

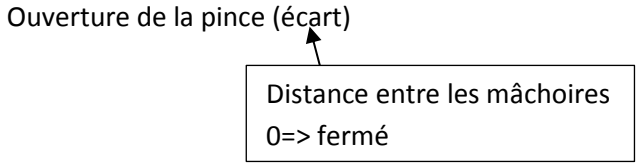
Exemple 1 :



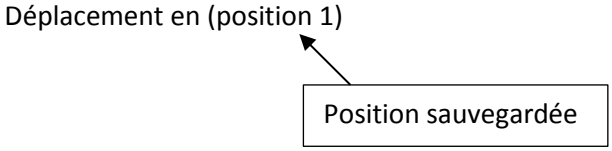
Exemple 2 :



Exemple 3 :



Exemple 4 :



- La position peut changer à volonté

Il faut pouvoir créer un programme complet à partir du Socket, pour déclencher tout une succession d'action sans être obligé de faire un contrôle permanent. En effet, les commandes Socket peuvent êtres préemptives, ce qui compliquerait de trop le control. Ces points sauvegardés doivent pouvoir être modifiés depuis le Socket. Nous devons comprendre comment accéder à tous les bus, E/S et autres paramètre de l'UR5.

En effet, à ce jour les commandes Socket sont assez mal documentées.

5. CONCLUSION ET PERSPECTIVE

Dans cet article, nous avons proposé différent moyen de contrôler des robots à distance avec l'utilisation de ROS et SocketTest. Le SocketTest a été concluant, nous avons pu aboutir à des résultats sur les déplacements de l'UR5. Le SocketTest est une interface directe de l'UR5 et qui permet pour tous les usages de développer facilement un programme sur tablette ou téléphone portable. Les résultats des trois hypothèses de déplacement a permis de trouvé des nouveaux moyens d'améliorations pour la précision de nos robots et d'amélioré nos système de préhension sur les bras robots (UR5).

Les résultats attendus doivent prouver que les bras robotiques sont là pour accomplir des opérations et surtout de pourvoir optimiser le temps de déplacements des robots dans une usine du future, et prouver qu'elle peut être faites le plus simplement possible.

Une méthodologie sera validée dans l'obtention de ces temps d'exécution et les critères de validation pertinente seront mis en avant.

Les perspectives sont de pouvoir déclencher un programme sur ROS et SocketTest.

Le mir 100 dispose d'une fonction de positionnement précis utilisant une forme de V comme présenté Figure 3 : schéma de positionnement du MIR 100. Nous souhaitons poser ces figure à différent endroits pour permette les positionnements inter robot. Les questions se posant et devant être résolu pour améliorer le système sont la précision de cette position. Nous serons certainement obligé d'utilise d'autres artifices de positionnement comme des capteurs ultrasons ou des caméras.

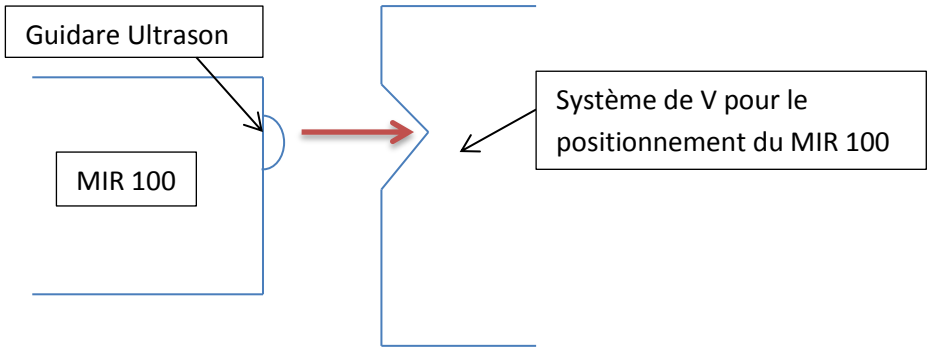


Figure 3 : schéma de positionnement du MIR 100

Le point 1 devra être résolu avec cette solution. Mais le point 3 est un peu plus difficile puisque les angles, bien que très précis, ne permettent pas d'avoir une précision millimétrique. Il faudra certainement positionner des caméras en bout de bras ou trouvés d'autre solution tel que Les préhenseurs électromagnétiques qui sont alimentés en tension continue qui va permettre d'avoir une marge de manœuvre pour prend l'objet. Pourquoi pas installer des capteurs capacitifs en bout de pince. L'UR5 prendre la pièce à bout de pince pour pouvoir l'attraper ce qui ne permet pas d'avoir un bon maintien de la pièce. Donc pourquoi pas prendre la pièce sur le côté avec la pince du MIR 100 pour avoir une meilleure prise, voir prendre que la pièce et non le support de pièce.

6. REFERENCES

[1] Researcher from Ecole Polytechnique Fédérale de Lausanne revealed the robo-arm in an article published today by IEEE transactions on robotics, the first journal in the field.

[2] Aleotti, J., & Caselli, S. (2006). Robust trajectory learning and approximation for robot programming by demonstration. *Robotics and Autonomous Systems*, *54*(5), 409-413.

[3] Alici, G., & Shirinzadeh, B. (2005). A systematic technique to estimate positioning errors for robot accuracy improvement using laser interferometry based sensing. *Mechanism and Machine Theory*, *40*(8), 879-906.

[4] https://fr.wikipedia.org/wiki/Robot_Operating_System

[5] Abhishek Jha , (Department of Mechanical Engineering, Visvesvaraya National Institute of Technology (VNIT), Nagpur, India)

[6] https://fr.wikipedia.org/wiki/Socket