Mise en pratique de compétences en robotique à partir d'un robot réel et de sa simulation

Sébastien Lengagne

Mise en pratique de compétences en robotique à partir d'un robot réel et de sa simulation.

Sébastien LENGAGNE
sebastien.lengagne@uca.fr
UCA - Clermont Auvergne INP/Polytech Clermont
Institut Pascal UMR6602 UCA/CNRS

***Résumé*—Dans le cadre des cours portant sur les robots manipulateurs, l'approche pédagogique s'est focalisée sur l'application à un système réel lors de séances de travaux pratiques et à la réalisation de challenge en autonomie en utilisant un environnement de simulation proche de l'environnement réel. La motivation de cette méthode est d'assurer la maitrise des concepts de base, (transformation entre repères, ...) lors de séance encadrée pour inciter les étudiants à appliquer ces concepts de manière autonome dans une application plus complexe.**

***Mots-clés*—robotique, localisation d'objets, modèle, commande, challenges**

I. Introduction

Cette mise en situation pédagogique s'inspire des challenges proposés dans le cadre du projet EUROC (https://cordis.europa.eu/project/id/608849) dont le but était de démontrer les compétences des équipes dans le cadre d'application de manipulation robotique industriel (pick-and-place, ...). Elle a pour public des étudiants de niveau BAC+4 en formation génie électrique ou équivalent.

L'objectif est de permettre aux étudiants de valider sur un robot réel les compéteces acquises et de préparer les séances et de finaliser leur apprentissage de façon autonome grâce à un environnement de simulation proche de celui utilisé en séance de Travaux Pratiques.

Cet article présente l'environnement utilisé et les outils dans la Section II puis les notions de robotique qui peuvent y être abordées comme les notions de repère dans la Section III, la modélisation en Section IV, la commande en Section V pour finir par la mise en place de challenge pour mettre en oeuvre les compétences en plannification/ordonnacement dans la Section VI.

II. Environnements et outils

A. Environnement réel

Dans le cadre de cet enseignement, les étudiants ont à disposition un robot Braccio (https://store.arduino.cc/products/tinkerkit-braccio-robot) équipé de plusieurs marqueurs associés à la base, d'une caméra USB et de plusieurs cubes à déplacer comme montrés sur la Figure 1. Le robot Braccio utilisé également en recherche [1] comporte cinq articulations (au sens robotique) et une pince actionnée. Ce robot mesure 52 cm pour un poids de 792 grammes, ce qui le rend utilisable en toute sécurité par les étudiants. Ses flexbilités et offset articulaires peuvent constituer un problème lors de la modélisation mais permettent de mettre en avant l'intérêt d'une commande visuelle comme montré dans la Section V. Plus de détail sur les objets de l'environnement sont donnés dans la Section VI.

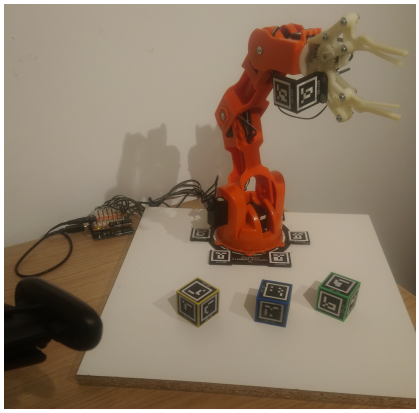


Fig. 1: Représentation de l'environnement réel utilisé en travaux pratiques.

Les fichiers des pièces imprimés en 3D, ainsi que les fichiers sources OpenSCAD sont disponibles sur https://github.com/lengagne/ArucoSTL.

B. Simulation

Les étudiants disposent également d'une version simulée de cet environnement sous Gazebo présentée sur la Figure 2 dont les fichiers sources sont disponibles sur https://github.com/lengagne/braccio_simulation. Cet environnement reprend les mêmes systèmes que l'environnement réel à l'exception de la caméra qui est libre (l'étudiant la déplace manuellement) en réel et qui est reliée à un robot virtuel en simulation. Dans le cas simulé, les imperfections du robot (flexibilité et offset articulaires) ne sont pas reproduites.

C. Outils à disposition

Dans ces deux environnements, les objets ainsi que la base du robot sont équipés de marqueurs de type Aruco [4]. L'information issue des marqueurs concerne l'identifiant du marqueur ainsi que sa pose dans le repère de la caméra comme montrée sur la Figure 3 et sera utilisée lors de la localisation des différents éléments.

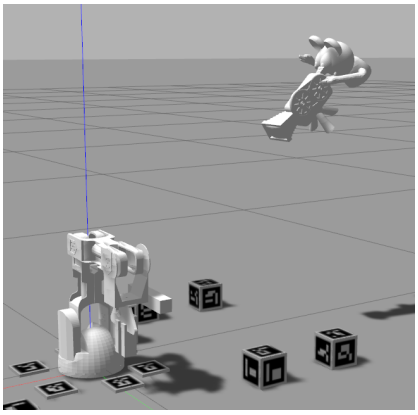


Fig. 2: Représentation de l'environnement simulé pour les travaux en autonomie.

Cet enseignement est à destination d'étudiants n'ayant pas de compétences en programmation orientée objet, un paquet ROS avec des codes pré remplis, un environnement de simulation RViz leur est mis à disposition et accessible via https://github.com/lengagne/braccio_etudiant.

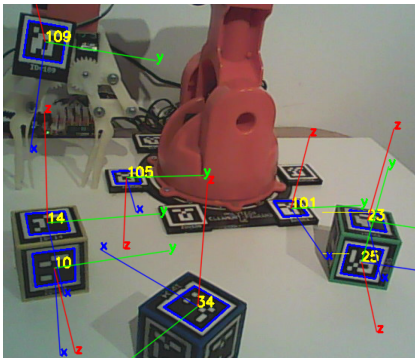


Fig. 3: Lecture des marquers aruco afin de récupérer leur identifiant et leur pose dans le repère caméra.

III. Gestion des repères - Localisation

La première tâche à réaliser est de localiser les objets dans le repère de référence (celui du robot). Pour cela, les étudiants utilisent les informations de la caméra qui sont traités par aruco pour en extraire l'identifiant et la pose de chaque marqueur dans le repère de la caméra. On obtient donc un ensemble de transformation pour chaque marqueur i visible par la caméra :

$$ {}^{cam}\mathbf{T}_i = \begin{bmatrix} {}^{cam}R_i & {}^{cam}P_i \\ 0 & 1 \end{bmatrix} \quad (1) $$

avec :

- ${}^{cam}\mathbf{T}_i$ la matrice de transformation aussi appelée matrice de passage ou pose,
- ${}^{cam}R_i \in \mathbb{R}^{3\times3}$ la matrice de rotation,
- ${}^{cam}P_i \in \mathbb{R}^{3\times1}$ le vecteur de position.

L'information de rotation étant donnée sous forme de quaternions, l'une des premières actions des étudiants est d'implémenter la convertion de cette information en matrice de rotation. Les actions suivantes nécessiteront d'implémenter l'inverse et la mutliplication entre deux matrices de transformation.

Comme indiqué sur la Figure 3, les marqueurs 100 à 106 sont fixés au support du robot, leur pose ${}^{w}\mathbf{T}_i$ est connue dans le repère monde w (world). Connaissant la pose d'un marqueur i dans le repère de la caméra, il est possible de calculer la pose de la caméra ${}^{w}\mathbf{T}_{cam}$ dans le repère monde :

$$ {}^{w}\mathbf{T}_{cam} = {}^{w}\mathbf{T}_i \times {}^{cam}\mathbf{T}_i{}^{-1} \quad (2) $$

Une fois la pose de la caméra connue dans le repère monde, en supposant un marqueur i associé à un objet o il est possible de localiser tous les objets dans le repère monde grâce à :

$$ {}^{w}\mathbf{T}_o = {}^{w}\mathbf{T}_{cam} \times {}^{cam}\mathbf{T}_i \times {}^{o}\mathbf{T}_i{}^{-1} \quad (3) $$

avec ${}^{o}\mathbf{T}_i$ la pose du marqueur i exprimée dans le repère de l'objet o.

Les informations de localisation des différents objets peuvent être visualisées sur Rviz (cf. Figure 4) après avoir été envoyés sur le topic $\backslash tf$.

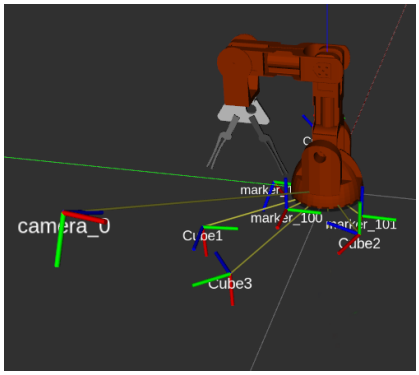


Fig. 4: Affichage de la localisation de la caméra et des différents objets via RViz correspondant à l'image de la Figure 3.

Cette étape a permis aux étudiants d'implémenter les opérations de base mettant en oeuvre des matrices de transformations et de les valider visuellement dans le cas de la localisation des différents éléments de l'environnement.

IV. Modélisation

Plusieurs types de modèles peuvent intervenir dans le cadre de systèmes robotiques (géométriques, cinématiques, dynamiques, direct ou inverse). Dans le cadre de cette application, les étudiants sont confrontés à l'implémentation des modèles géométriques directs et inverses.

A. Modèle Géométrique Direct (MGD)

Le modèle géométrique direct (MGD) permet de calculer la pose de l'effecteur (ici la pince) dans le repère monde ${}^{W}\mathbf{T}_{eff}$ en fonction de la position des différents moteurs (appelés variables articulaires) q

$$ {}^{W}\mathbf{T}_{eff} = MGD(q) \quad (4) $$

Ce modèle peut être calculé de manière itérative en partant du repère de référence et en calculant la pose de chaque repère (associé à chaque corps du robot) jusqu'à l'effecteur. Plusieurs méthodes existent pour construire ce modèle, elles se différentient principalement par la définition des repères liés à chaque corps :

- Denavit Hartenberg (DH) : cette méthode définit les repères de manière à passer d'un repère à un repère suivant en utilisant uniquement quatre paramètres (alors que 6 sont nécessaires dans un cas général) [2],
- Denavit Hartenberg modifiée par Khalil et Kleinfinger (DH-KK) reprend le même principe mais permet de diminuer le nombre d'opérations mathématiques grâce à une définition différentes des repères [6],
- Unified Robot Description Format (URDF) ne contraint pas la définition des repères, puisqu'elle définit les six paramètres nécessaires. L'utilisation de la programmation templatée [8] permet de ne considérer que les opérations utiles à la modélisation (la multiplication par 1 ou par 0 est résolu à la compilation, pas à l'éxécution).

Ces trois méthodes sont présentées en cours, les étudiants sont libres d'implémenter celle qui leur parait la plus pertinente (avec une préférence pour l'URDF qui paraît plus intuitif et moins sujet aux erreurs). Afin de valider leur modèle, une représentation du robot et du repère de l'effecteur sont affichées sur RViz avec une fenêtre permettant de bouger les variables articulaires afin de vérifier que la pose calculée de leur effecteur demeure bien entre les pinces du robot comme montré sur la Figure 5.

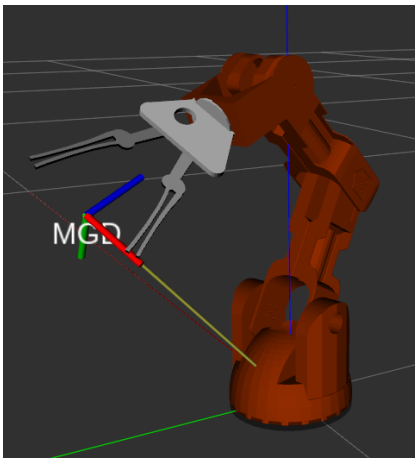


Fig. 5: Affichage de la pose de l'effecteur calculé via la modèle géométrique direct sous RViz.

Lors de la première séance de TP (4h) sur ce point, les étudiants sont assez guidés. Par la suite, l'objectif est de les laisser se confronter aux notions avancées en robotique, comme par exemple le modèle géométrique inverse.

B. Modèle Géométrique Inverse (MGI)

Le modèle géométrique inverse (MGI) permet de calculer les valeurs articulaires en fonction d'une pose désirée.

$$q = MGI({}^{W}\mathbf{T}_{eff}) \tag{5}$$

Connaissant la pose des différents objets, ce modèle permettrait de calculer les valeurs articulaires permettant au robot de placer le repère de son effecteur sur le repère lié à un objet pour s'en saisir. La résolution de ce problème amène les étudiants à considérer le robot comme étant sous-actionné ou redondant.

1) Robot sous-actionné: En effet, si on définit la pose de l'effecteur via trois positions et trois rotations, les cinq articulations du robot ne permettent pas de trouver une solution dans tous les cas. Cette limitations est liée à la structure du robot ainsi qu'au nombre d'articulation trop faible.

2) Robot redondant: Cependant, l'orientation de la pince n'a pas réellement d'importance pour la saisie de l'objet. Une solution serait de ne considérer que la position de l'effecteur afin de trouver les valeurs articulaires permettant la saisie de l'objet. Cependant dans ce cas, les cinq degrés de liberté (articulation) sont redondants pour satisfaire les trois équations de position, ce qui a pour effet d'avoir une infinité de solution possible. L'étudiant doit donc lever cette indetermination en considérant de nouvelles équations et en évaluant l'impact de ces choix sur les capacités du robot. Par exemple, considèrer de garder la pince verticale dimunera grandement l'espace atteignable du robot.

3) Méthode implémentée: Quelque soit les choix faits par l'étudiant l'implémentation de ce modèle peut se faire par différente méthode : méthode de Paul [5], résolution d'un problème d'optimisation [3] ou de manière itérative en simulant une loi de contrôle permettant de se rapprocher de la pose désirée.

La validation de l'implémentation peut se faire en visualisant sur RViz. Cependant, lors de validation réelle, les imperfections du robot peuvent amener à une pose de l'effecteur suffisament différente pour ne pas permettre la saisie de l'objet. Dans ce cas une méthode de commande basée sur les informations réelles via la caméra doit être mise en place.

V. Commande

A. Utilisation de la Jacobienne

Une loi de commande va être mise en place pour compenser les différentes erreurs de modélisation ou de localisation. Une des méthodes les plus simples et d'utiliser la Jacobienne et sa pseudo-inverse afin de définir les vitesses angulaires qui vont amener le repère de l'effecteur à se rapprocher de la pose désirée.

$$\dot{q} = J^{+}(q)(\mathbf{T}_{des} - {}^{W}\mathbf{T}_{eff}) \tag{6}$$

avec :

- $J(q)$ la jacobienne du robot vis à vis de l'effecteur qui peut être obtenue en dérivant le modèle géométrique direct par rapport aux variables articulaires,
- $J^{+}(q)$ la pseudo inverse de Moore Penrose,

- $\mathbf{T}_{des}$ la pose désirée de l'effecteur.

Comme dans le cas du modèle géométrique inverse, la structure du robot ne permet pas de considérer les six composantes de la pose, dans notre cas il est plus judicieux de ne considérer que la position de l'effecteur.

B. Position de l'effeteur

L'équation (6) met en oeuvre la pose de l'effecteur pour permettre un asservissement de sa position. Pour obtenir cette pose, il serait possible d'utiliser le modèle géométrique direct à partir des variables articulaires q. Même si cela n'a pas d'utilité en pratique car les erreurs de modélisation seraient présentes, cette première étape permet une validation en simulation du comportement de la loi de commande. Par la suite, la pose réelle de l'effecteur sera obtenu grâce à des marqueurs reliés à l'effecteur comme le montre la Figure 6.

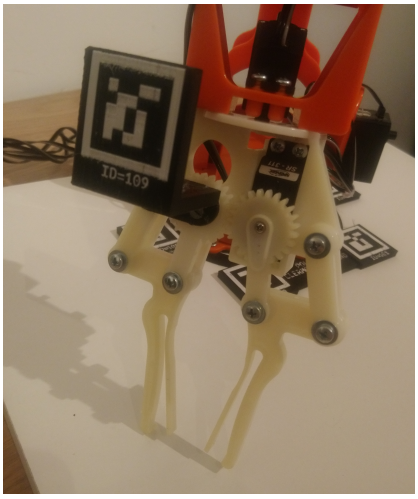


Fig. 6: Représentation des marqueurs liés à l'effecteur afin de mesurer sa pose réelle.

C. Mise en place d'une pile de tâches

Une amélioration de l'équation (6) est possible en considérant un ensemble de tâches comme défini dans [7], la plus importante restant la position de l'effecteur. D'autres tâches de priorité moindre pourraient être relatives à l'orientation de la pince. Des améliorations de cette méthode pourraient également être mise en place pour prendre en compte les contraintes de collision avec le sol ou d'autres éléments de l'environnement.

Une fois ces étapes réalisés, le robot est capable de positionner son effecteur à différentes position de manière précises en se basant sur un modèle géométrique et/ou une loi de commande afin de s'affranchir des perturbations ou d'erreurs de modélisation. L'accomplissement de mouvement complexes (de type pick-and-place) ne constitue alors qu'un ordonnancement de plusieurs actions de positionnement de l'effecteur lié à une ouverture ou fermeture de sa pince.

VI. Challenges

Pour finaliser l'apprentissage de ces compétences en robotique, les étudiants doivent réaliser plusieurs tâches définies sous la forme de challenges, principalement en simulation. Les étudiants ont accès à des environnements pour chaque challenge et seront évalués sur d'autres environnements afin d'éviter toute solution "hard-codée".

A. Objets de l'environnement

Les challenges mettent en oeuvre des cubes comportant un marqueur Aruco sur chaque face comme le montre la Figure 7

Fig. 7: Représentation d'un cube dans l'environnement de simulation

Au total 18 cubes seront utilisés. Chaque cube est défini par son identifiant i (de 1 à 9 et de 11 à 20) et comporte les marqueurs aruco numéro $i \times 10 + 0$ à $i \times 10 + 5$. Il y a deux boîtes utilisées pour les challenges présentées sur la Figure 8.

- La boite 1 est un cube de 15 cm de côté portant les marqueurs 1 à 4 sur chacun de ces côtés.
- La boite 2 est un cube de 10 cm de côté portant les marqueurs 5 à 8 sur chacun de ces côtés.

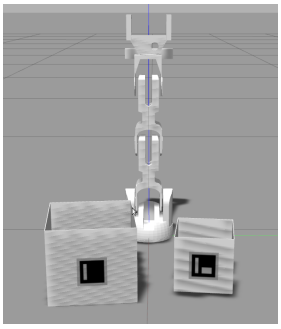


Fig. 8: Représentation des boites (boite 1 à gauche, boite 2 à droite) dans l'environnement de simulation.

Les challenges commencent par reprendre les notions vues précedemment puis incitent l'étudiant à aller plus loin. Un exemple des environnements de base est présentés sur la Figure 9. Pour chaque challenges, plusieurs environnements sont disponibles sur https://github.com/lengagne/braccio_simulation. La durée maximale de la simulation pour la résolution de chaque challenge est limitée à deux minutes.

B. Challenge 1 : Localiser les cubes

Le premier challenge consiste à identifier et à localiser dans le repère de référence les cubes présents autour du robot. Pour cela avant la fin du challenge les identifiants des cubes (id) présents et leur localisaiton($x, y, z, roll, pitch, yaw$) doivent être envoyés sur un topic dédié. Cette localisation est vue en séance avec l'enseignement, l'objectif est de vérifier que l'étudiant sait redeployer le travail réalisé précedemment. De plus elle permet de valider le code de commande de la caméra qui n'a pas été vue en pratique.

C. Challenge 2 : faire tomber une tour

L'environnement est composé de plusieurs tours composées de cubes empilés. L'objectif est de toucher les tours formées par les cubes afin de les faire tomber. Cela permet de valider que l'étudiant sait envoyé des commandes de mouvements au robot.

D. Challenge 3 : Pick and place

L'objectif de ce challenge est de placer tous les cubes présents dans l'environnement dans la boîte 1 ou dans la boîte 2 (qui est plus petite). Cette première tâche nécessitant un ordonnancement permet de valider le concept de pick and place.

E. Challenge 4 : inversion de cube

L'objectif est d'inverser la position des cubes de même unité (1 et 11, 2 et 12, ...). Si un cube est le seul de la scène à avoir cette unité, il devra rester à sa place.

F. Challenge 5 : Empiler les objets

L'objectif est d'empiler les cubes de même dizaine (1 à 9 sur une pile et 11 à 19 sur une autre pile) sans que l'ordre ne soit important. Cette tâche nécessite de la précision dans le contrôle et une étape de vérification de bon alignement des cubes pour garantir l'équilibre des tours.

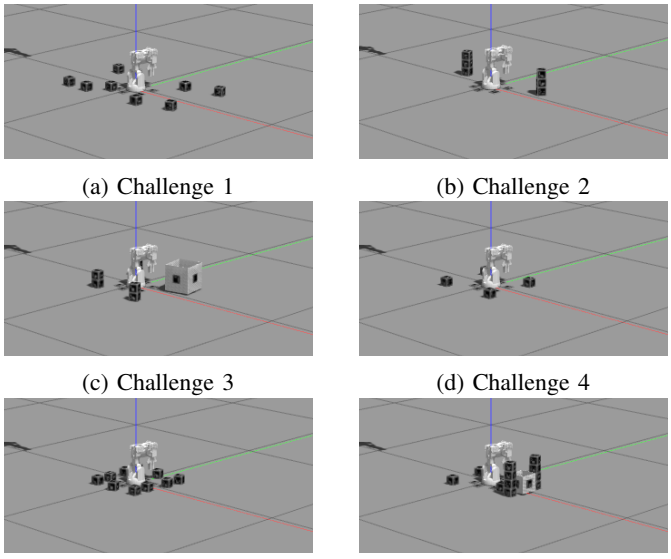
G. Challenge 6 : Evitement de collision

L'objectif est de placer les cubes entre 1 et 9 présents dans l'environnement dans la boite 2, sans faire tomber les piles constituées des cubes 11 à 19. Cette dernière tâche nécessite une planification plus fine prenant en compte tous les corps du robot.

Il est possible de prévoir de nouveaux challenges en combinant les 6 challenges existants ou d'en créer des nouveaux.

VII. Perspectives et conclusion

Cet article présente un ensemble d'outils mis en place dans le cadre d'un enseignement en robotique. Les fichiers pour la reproduction en réel ou en simulation sont disponibles. Actuellement, les TPs ont été réalisés sous Ubuntu 18 avec la version melodic de ROS1. Une évolution vers ROS2 devra être menée. Il est également envisagé d'ajouter des environnements plus proche d'environnement industriel mettant en oeuvre, par exemple, un tapis roulant qui ferait arriver les cubes ou des tâches co-manipulation.



(a) Challenge 1 (b) Challenge 2

(c) Challenge 3 (d) Challenge 4

(e) Challenge 5 (f) Challenge 6

Fig. 9: Représentation des 6 challenges proposés aux étudiants.

Le principal retour des élèves concerne une prise en main difficile lié à un manque d'expertise en programmation. Pour palier ce point, des tutoriels de programmation en C++ leur seront fournis. Ce format permet de valider pas à pas les compétences visées sur le robot réel et de permettre aux étudiants d'avancer entre les séances grâce à l'environnement de simulation.

References

[1] Cao, Y., Wang, T., Qian, X., Rao, P. S., Wadhawan, M., Huo, K., and Ramani, K. Ghostar: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2019), UIST '19, Association for Computing Machinery, p. 521–534.

[2] Denavit, J., and Hartenberg, R. S. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics 22*, 2 (06 2021), 215–221.

[3] Escande, A., Mansard, N., and Wieber, P.-B. Fast Resolution of Hierarchized Inverse Kinematics with Inequality Constraints. In *ICRA 2010 - IEEE International Conference on Robotics and Automation* (Anchorage, United States, May 2010), pp. 3733–3738.

[4] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., and Marín-Jiménez, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition 47*, 6 (2014), 2280–2292.

[5] Khalil, W., and Dombre, E. *Modeling, Identification & Control of Robots*, 3 ed. Hermes Sciences Europe, march 2002.

[6] Khalil, W., and Kleinfinger, J. A new geometric notation for open and closed-loop robots. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation* (1986), vol. 3, pp. 1174–1179.

[7] Mansard, N., Aycard, O., and Koike, C. Hierarchy of behaviours. In *IEEE Int. Conf. on Robotics and Biomimetics, ROBIO'05* (Hong-Kong, China, July 2005), pp. 583–588.

[8] Naveau, M., Carpentier, J., Barthelemy, S., Stasse, O., and Souères, P. Metapod template meta-programming applied to dynamics: Cop-com trajectories filtering. In *2014 IEEE-RAS International Conference on Humanoid Robots* (11 2014), vol. 2015.