Walking Machine @Home
Rapport de projet 2018

Jeffrey Cousineau and Philippe La Madeleine

École de Technologie Supérieure
1100 rue Notre-Dame Ouest, Montreal, QC, Canada H3C 1K3
http://walkingmachine.ca, walking@ens.etsmtl.ca,
https://github.com/WalkingMachine

Résumé Ce papier donne des détails à propos du projet de l'équipe Walking Machine de l'école de technologie supérieure, participant à la compétition Robocup@Home. Leur prochaine compétition se déroulera à Montréal, dans leur ville natale. Le robot du club Walking Machine se nomme S.A.R.A. pour "Système d'assistance robotique autonome". Ce robot est entièrement conçu par le regroupement de passionnés et est principalement composé d'élève du 1er cycle en génie. Leur robot est utilisé pour l'interaction humaine et l'assistance personnelle. Ce document démontrera les différentes avancées technologiques apportées dans la dernière année au niveau logiciel, électrique et mécanique.

1 Introduction

L'équipe de Walking Machine est une jeune équipe de Montréal et composée de jeunes étudiants dans le domaine du génie mécanique, électrique et logiciel. Elle a beaucoup travaillé au niveau de l'amélioration du robot pour la prochaine année afin de participer pour une troisième fois à la compétition Robocup@Home. Après quelques années d'expériences, il y a eu un très grand apprentissage, ce qui en résulte en de de belles améliorations qui mèneront à de meilleurs résultats, principalement du côté logiciel. Dans le passé, l'équipe a participé à plusieurs compétitions comme la Eurobot mais, a décidé de changer afin de faire face à un plus grand défi et ainsi, d'avoir l'opportunité d'apporter une vague d'innovation dans le monde scientifique entourant la robotique.

S.A.R.A. fut conçue pour une interaction humain-robot polyvalente et une navigation efficace. La plateforme robotique est équipée de roue mecanum et de moteurs maxon alimentés par des drives Roboteq. Elle a également un bras permettant d'immiter un bras humain ainsi que des capteurs pour la communication et la navigation. Notre équipe a développé une certaine expérience dans la reconnaissance de visages et d'objets en plus de la navigation utilisant un laser 2D. Ces différents modules sont interfacés via ROS (Robot Operating System).

2 Améliorations électriques et mécaniques

2.1 Électrique

Comme amélioration cette année du côté électrique de notre robot, nous avons mis beaucoup d'efforts sur l'organisation de l'élaboration du système en le rendant plus clair et sécuritaire. Nous avons donc ajouté des protections pour chaque sous-système électrique puisqu'il est ainsi beaucoup plus facile de changer un fusible qu'un circuit complet.

Un autre gros changement pour nous fut le système de batterie. Tout juste avant de partir pour notre compétition au Japon en juillet 2017, nous avons changé notre ancienne batterie LiPo maison par des batteries de perceuse. Ce changement a apporté beaucoup plus de points positifs que prévus. Tout d'abord, cela a simplifié l'exportation internationale de notre robot et nous permettait d'emmener les batteries dans nos bagages personnels. Le deuxième impact majeur fut le sentiment de sécurité qui accompagnait ce type de batterie. Il est beaucoup facile de les charger et sécuritaire puisque ce type de batterie possède possède leur propre chargeur adapté avec des circuits de protection. De plus, notre système à deux batteries instauré dans notre robot nous donne la possibilité d'enlever une des deux batteries sans avoir à éteindre le système complet.

2.2 Mécanique

Quelques améliorations mécaniques furent également complétées durant l'année. Il y a principalement deux aspects qui furent repensés, soit le bras ainsi que la base.

Tout d'abord, l'année passée, notre bras possédait cinq degrés de liberté, ce qui rendait la planification de trajectoire beaucoup plus complexe et limitait grandement les mouvements. Nous avons donc décidé de rajouter deux degrés de liberté grâce à deux servo-moteur Dynamixel. Depuis cette modification, nous avons des résultats grandement améliorés et une plus grande liberté de mouvement. Cela a également comme impact de rallonger le bras ce qui nous donne un plus grand rayon d'action.

L'autre aspect principal qui fut amélioré était la base de notre robot. Suite à notre première compétition en Allemagne, nous avons observé que la base était beaucoup trop large, ce qui causait quelques problèmes au niveau de la navigation autonome près des portes. En se basant sur ces observations, nous décidé de réduire la largeur ce qui a eu pour effet d'améliorer grandement les capacités en navigation.

Figure 1. Bras à 7 degrés de liberté

3 Logiciel

3.1 Planification des tâches de haut-niveau

Pour notre planification des tâches que notre plateforme pouvait accomplir, nous avons utilisé un module développé par une équipe de robotique participant à la compétition Darpa Robotics Challenge. Ce logiciel nommé FlexBe[4], pour Flexible Bahavior, est composé d'une interface de style programmation par blocs, il est donc facile et clair d'implémenter les différents scénarios pour la compétition.

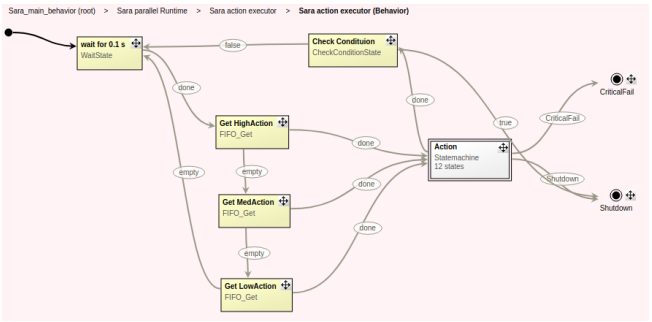

Figure 2. Représentation de FlexBe

Par contre, nous n'utilisons pas tout simplement FlexBe, nous devons programmer nos propres états en utilisant l'API en python. Pour simplifier notre travail, nous avons commencé par diviser tous les scénarios de base demandé lors de la compétition Robocup@Home en petites actions reproductibles. Nous avons ensuite repérées ceux pouvant être accomplies par notre robot et les avons transformée en état (ex : mouvement du bras à une position X). Ces blocs d'état peuvent ensuite être assemblés ensemble pour former une hierachie plus élevée que l'on nomme Actions. Nous assemblons finalement ces actions en ActionWrapper qui permettront d'interfacer les actions générale de notre robot avec la détection vocale ainsi que le traitement du language. Ces différents modules interpêtent ainsi le language et le transforme en action réalisable par le robot. Cette structure récursive nous permet donc un développement beaucoup plus rapide pour chaque tâches nécessaires.

3.2 Natural language processing

To analyze the detected speech, we rely on a software develop by the Semantic analytic group from the University of

Roma and the Laboratory of Cognitive Cooperating Robots at Sapienza University of Rome. This speech analyzer, named LU4R[1] for "Language Understanding For Robots", is composed of a server developed in Java which takes as an input the detected sentence and the semantic environment surrounding the robot.

This server communicates through a REST service which gives it the possibility to be compliant with all kind of platform. All you have to do is to launch the server locally which is compiled through a .jar file. Again, this gives the opportunity to use this software on every platform, whether you're on Windows, Linux or Mac. This software gives you the possibility to get different output representation. We choose the amr representation since it was the easiest one to understand and to implement.

We decided to build our own ROS wrapper, lu4r_ros to better interface it with our task planning approach. We first translate the answer given by LU4R into the simple format, we call ActionForms. The ActionForms contains an action followed by all of its possible parameters as identified in the FrameNet Index of Lexical Units The ActionWrappers are then fed into a FIFO based priority manager and finally send to our task planner.

What is also interesting about LU4R, it's that it will use the semantic mapping in it's analyzing process. All you have to do is to provide the correct pose for every object in the robot's environment. You can also precise various synonym for every object to get a better understanding of the inputted sentence.

3.3 Object recognition

As a beginning team, we are still exploring various solution around the object recognition problem. Our first plan was to use the object recognition kitchen package from Willow Garage. But after using it in competition, we realized that the performance and the easiness to use wasn't the approach we were looking for. As an alternative, we start using the YOLO[3] (You Only Look Once) ros package.

YOLO is a real-time object detection. It does not only detect various object but it also predicts the bounding boxes of the detected object. It uses a single neural network which is applied to the image. Multiple regions are then created and are used to predict the bounding boxes. Each of them also contains the predicted probability which is used to filter the predicted objects. The advantage of this system is that it can detect multiple objects in a real-time scenario.

We use the ROS package to make our job easier since this gave us the possibility to directly get the recognized objects output into a ROS topic. We can also get the bounding box for each detected object. First step we had to do was

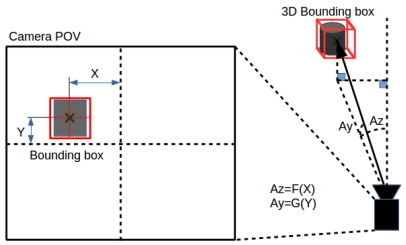

Figure 3. 2D bounding box to 3D grasping pose, current technique

to transform those 2D bounding boxes in 3D to get a specific pose according to our robot.

For the moment, we created the package wm_frame_to_box to approximate the object pose with the depth point at the center of the bounding box. Even though it can have flaws, this technique has also proven to be largely sufficient for most of our applications and most importantly, it uses way less processing power than the full 3D pattern matching we used before. This allows us to do real-time object positioning, a capability we are proud of.

Afterward, to get better results and a better pose, we plan to subtract the point cloud according to the bounding boxes. We will then use it with the point-cloud segmentation from the PCL library to extract the specific object and send it to a grasp identification package like haf_grasping. This technique, compared to what we are actually using would give us the possibility to grab a much wider variety of object.

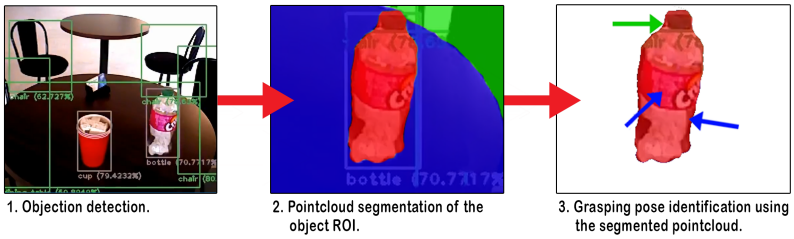

Figure 4. 2D bounding box to 3D grasping pose, future technique

For the moment, we are using the YOLO model but, we are looking forward to training our own dataset, just like we would do in competition. This is new

for us but we have all the tools we need to overcome this.

3.4 Navigation

En plus de notre système de navigation précédemment utilisé, cette année nous utilisons le module pointcloud_to_lasercan comme solution plus légère afin de faire de l'évitement d'objets en 3D. Cela nous assure une navigation beaucoup plus sécuritaire puisqu'en utilisant seulement une navigation 2D à l'aide d'un lidar, il serait possible que le robot entre en collision avec une table puisqu'il ne verrait que les pattes de celle-ci. Un autre avantage de cette implementation est qu'il est possible de spécifier la hauteur maximum au niveau des collisions. Ainsi, nous évitons également les possibles collisions avec la tête de notre robot.

3.5 Représentation de l'environnement

Comme nouveauté cette année, nous avons implémenté notre propre solution afin de représenter l'environnement entourant le robot d'une façon simple pour tout le monde. Notre solution que nous avons nommée wonderland est un système agnostique dans le même genre que LU4R présenté plus tôt. Il est composé d'un serveur recevant une requête HTTP basée sur un API développé par l'équipe.

La première choses requise pour commencer à utiliser cette plateforme, est l'action de populer la base de donnée. Cela peut être fait de façon manuelle en indiquant la position de chaque objet connu, mais cela peut être fait par le robot à l'aide de requête POST. Il est également possible de spécifier une pièce avec une position relative à celle-ci. Lorsque les différentes informations sont insérées, il est possible de faire une requête GET sur l'adresse définissant le serveur de wonderland (ex : http ://wonderland :8000/object). Cette dernière retournera la liste complète des objets contenue dans la base de donnée. Il est également possible de filtrer les demandes en rajoutant par exemple une couleur ou une position comme paramètre.

Puisque la base de donnée est hébergée sur un serveur, il est possible d'y accéder de puis n'importe où. Il est possible de l'exporter sur un autre ordinateur, de la mettre en infonuagique ou seulement de l'installer sur l'ordinateur même du robot. Cela donne également la possibilité d'avoir une connaissance dynamique, ce qui signifie que le robot peut mettre à jour ses connaissances de l'environnement en temps réel.

4 Conclusions et projets futures

Comme vous pouvez le constater, malgré que nous soyons une équipes composée principalement d'étudiants de premier cycle, nous sommes tout de même capable de rivaliser avec les autres équipes composées de laboratoire de recherche. Nous avons recemment mis beaucoup d'effort afin d'avoir une plateforme plus stable et de réduire les différents bogues présents, nous permettant donc d'avancer dans la bonne direction.

Avec notre nouveau système de batteries interchangeables, notre détection d'objets et l'inteprétation du language naturel, notre robot devient est devenu une plateforme autonome complètement fontionnelle, nous permettant donc de nous concentrer sur le coeur des challenges de la compétition.

Spécifications matérielles

SARA	
Base	Base avec roues holonomiques mecanums
Bras	7 DoF custom arm made of Kinova motors
Cou	Tilt and pan unit using two Dynamixel MX-64R servo actuator
Tête	Custom head made of RGB neopixels leds and Asus Xtion Pro
Pince robotique	Robotiq 2 doigts 140mm
Dimensions	Base : 0,61m. X 0,77m. Height : 1,68m.
Poids	∼60kg
Capteurs additionnels	Hokuyo UTM-30LX on base
Microphone	Microphone Rode
Batteries	2x 20V Dewalt drill batteries 5aH
Ordinateur	1x Lenovo p50 avec 32GB RAM et nVidia Quadro M2000 4GB, 1x Raspberry Pi 3

Table 1. Robot's hardware description

Robot's Software Description

For our robot we are using the following software :

— Platform : Robotic Operating System (ROS) Kinetic on Ubuntu 16.04
— Navigation, localization and mapping : Gmapping, AMCL, pointcloud_to_laserscan
— Face recognition : People
— Speech recognition : Google Speech API
— Speech comprehension : LU4R, lu4r_ros
— Speech generation : Svoxpico
— Object recognition : Darknet with YOLO v2
— Arm control : MoveIt and Kinova API
— Task executor : Flexbe
— World reprensentation : Wonderland

Figure 5. Robot SARA

Membres de l'équipe

Jeffrey Cousineau, Philippe La Madeleine, Maxime St-Pierre, Nathalie Connolly, Jimmy Poirier, Léonore Jean-François, Samuel Otis, Redouane Laref, Louis-Charle Labarre, Lucas Maurice, Nicolas Nadeau, Simon Landry, Cheuk Fai Shum, Veronica Romero Rosales, Nicolas Bernatchez, Quentin Gaillot, Raphael Duchaine, Jean-Frederic Boivin

Références

1. LU4R Project - adaptive spoken Language Understanding For Robots.
2. Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.
3. Joseph Redmon and Ali Farhadi. Yolo9000 : Better, faster, stronger. *arXiv preprint arXiv :1612.08242*, 2016.
4. Philipp Schillinger, Stefan Kohlbrecher, and Oskar von Stryk. Human-Robot Collaborative High-Level Control with an Application to Rescue Robotics. In *IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, May 2016.
5. Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2016.
6. Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA, 2004.
7. Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.
8. Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.
9. Stephen J. Wright. Coordinate descent algorithms. *Math. Program.*, 151(1) :3–34, June 2015.