CoreBot M : Le robot de la Team CoreBots préparé pour l'édition 2011 du défi Carotte

Bruno Steux, Laurent Bouraoui, Sylvain Thorel, Louis Benazet

CoreBot M : Le robot de la Team CoreBots préparé pour l'édition 2011 du défi Carotte

Bruno STEUX, Laurent BOURAOUI, Sylvain THOREL, Mines ParisTech
Louis BENAZET, Intempora SA

***Abstract*—La deuxième édition du défi CAROTTE, organisé par l'ANR et la DGA, aura lieu fin juin à Bourges. Son objectif : vérifier la capacité de petits robots terrestres pour des missions de reconnaissance en environnement intérieur. Après avoir remporté l'édition 2010, la Team CoreBots présente son nouveau robot dédié à l'édition 2011. Sans en révéler tous les secrets, ce papier expose les grandes lignes de notre architecture et de notre stratégie.**

INTRODUCTION

Afin de répondre aux exigences du nouveau règlement et aux évolutions technologiques - notamment l'apparition de la Kinect -, la Team CoreBots présentera en 2011 un robot profondément modifié, appelé *CoreBot M*. En voici les principaux points marquants :

- L'adoption d'une toute nouvelle plateforme mobile à 6 roues (cf. figure 1). Comme l'année dernière, nous n'avons pas retenu une plateforme dédiée à une utilisation en intérieur, mais une plateforme très générique. Capable de franchir des marches ou de rouler dans le gravier ou l'herbe haute, ce robot 6 roues se rapproche d'un produit fini. Le *CoreBot M*, architecturé autour d'une carte mère à base de Core i7 faisant tourner Ubuntu 10.04LTS, intègre un capteur laser Hokuyo UTM30LX, une Kinect, une centrale inertielle 6 axes VectorNav (utilisée pour détecter les pentes), et un capteur ultrasons très directif permettant d'éviter vitres et autres miroirs...
- L'adoption d'un nouveau capteur : la Kinect, utilisé pour la reconnaissance des objets comme pour l'évitement des obstacles proches. Il permet aussi de reconstruire en couleur l'environnement 3D du robot. Contrairement à l'édition 2010 où le robot devait s'arrêter pour effectuer un Scan 3d, les scans sont cette année obtenus à la volée sans arrêter le robot, tout en étant texturés. Par ailleurs, la Kinect nous permet aussi de détecter tous les objets proches à éviter, y compris les tables que nous avions bien du mal à éviter l'année passée (nous tentions de passer dessous, entrainant le renversement du robot...). Le SLAM continue à s'appuyer sur le laser Hokuyo UTM30LX, qui offre une précision et une portée très supérieure à la Kinect et surtout une converture de 270°, très adaptée à un robot à la vitesse de rotation importante.
- Un logiciel embarqué présentant de grosses évolutions : une nouvelle version du middleware *Cables*, un nouvel algorithme de SLAM : *CoreSLAM2*, un nouvel algorithme de path planning : *CoreControl2*, et un nouvel algorithme de reconnaissance d'objets 3D basé sur AdaBoost : *Core3DLearner* et *Core3DAnalyzer*.
- Et une toute nouvelle stratégie d'exploration optimisée pour le défi et cherchant à examiner chacun des recoins de l'arène : *CoreCarotte2*.

Figure 1. Le robot CoreBot M de la Team CoreBots

Comme l'année dernière, la spécificité de notre solution réside dans l'utilisation exclusive d'éléments développés en interne de l'équipe : de la plateforme aux algorithmes, en passant par le middleware Cables, Corebot M s'appuie des solutions maîtrisées intégralement. Seuls les capteurs, la carte mère et le système d'exploitation sont des apports externes.

DE L'INTÉGRATION DE LA KINECT

L'intégration de la Kinect a fait l'objet d'un soin particulier. Elle s'appuie sur la réalisation de 3 composants *Cables* dédiés :

- Le *kinect_server*, qui délivre les images de profondeur et les images RGB ou YUV422 acquises par la Kinect.
- Le *kinect_3d_server*, qui associe l'image couleurs avec l'image de prondeur et produit un nuage de points texturé (cf. figure 2). Ce composant s'appuie sur les techniques

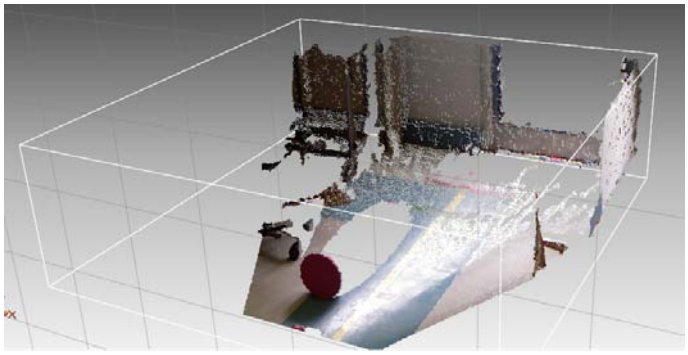

Figure 2. Un nuage de points acquis avec la Kinect embarquée sur CoreBot M

de calibration développées par Nicola Burus et Konolige pour ROS. Ce composant est configurable en terme de vitesse d'acquisition et de résolution. Il permet aussi à travers le service "kinect.save" d'enregistrer des nuages de points.

- Le *kinect_voxel_server*, qui effectue la voxelisation du nuage de points. Il repositionne ce nuage de points dans un repère global, utilisant à la fois la position extrinsèque de la kinect sur le robot, et la position du robot fournie par le SLAM. Le nuage de points trié et indexé fourni en sortie est prêt à être exploité pour la segmentation des objets, du sol et des murs. Il est également directement utilisé pour la reconstruction 3d de l'ensemble de l'environnement, par simple accumulation. Une somme verticale des voxels est également produite. C'est cette somme, identifiant les objets proches du robot, qui est envoyée à CoreControl2 pour leur prise en compte dans le calcul de trajectoire et donc l'évitement d'obstacles.

Ces différents composants communiquent ensemble grâce à la fonctionnalité de mémoire partagée de *Cables 0.8*, c'est à dire à coût quasi nul bien que chaque composant tourne dans son propre processus.

Le positionnement précis des données de la Kinect par rapport aux données issues du SLAM est assurée par une synchronisation offline entre les capteurs laser Hokuyo (utilisé pour le SLAM) et la Kinect. Toutes les données sont datées et synchronisées au sein du système par Cables (nouvelle fonction *cables_sync*).

CoreSLAM2 et CoreControl2

CoreSLAM2 et *CoreControl2* ont été intégralement réécrit cette année. *CoreSLAM2* a été conçu de manière beaucoup plus évolutive que CoreSLAM1, notamment du fait de son architecture client / serveur. Elle permet de réaliser des clients distants récupérant la carte en temps-réel, par transmission de différences de cartes. Par ailleurs, *CoreSLAM2* est beaucoup plus précis et performant que son prédécesseur, exploitant notamment lieux les différents coeurs d'un processeur. Le filtrage d'obstacles temporel permet en particulier de faire disparaître de la carte des obstacles n'ont été vus que brièvement. Par exemple, une personne qui se trouve brièvement dans le champ du laser sera effacée de la carte même si elle se trouve en dehors du champ du laser.

Par ailleurs, *CoreSLAM2* fait maintenant du SLAM 2D1/2 : il exploite les données de la centrale inertielle du robot pour évoluer correctement sur sol non plat, ceci afin de prendre en compte l'évolution du règlement (présence de pentes et de zones surélevées). A noter que les données de la centrale inertielle sont synchronisées par notre laser, toujours grâce à notre middléware *Cables* et une synchronisation off-line.

CoreSLAM2 conserve son principal avantage : il peut être exploité sans odométrie, ce qui élimine le problème de la dérive lié à la désynchonisation entre données odométriques et données laser. Comme l'année dernière, une carte de distance hexagonale est produite en temps réel et est exploitée par le générateur de trajectoire.

Le générateur de trajectoire et contrôleur de robots *CoreControl2* a été entièrement revu. La version de l'année dernière était conçue pour un robot de forme ronde, alors que cette année notre robot a une forme allongée : il ne passe pas latéralement dans une ouverture de porte, et ce point doit être pris en compte par le contrôleur. Nous avons donc réalisé un tout nouveau générateur de trajectoire, qui calcule la trajectoire d'un point A à un point B, en considérant l'angle d'arrivée. La génération de trajectoire exploite un algorithme *A** qui explore l'espace *(x, y, θ)*, cherchant la trajectoire qui minimise la distance parcourue, tout en assurant le miminum de rotations et conservant une bonne distance aux obstacles. Un tel algorithme est très coûteux, notamment en mémoire, car l'espace de recherche est immense. Afin de la faire tourner en temps-réel sur le robot, nous avons appliqué quatre optimisations :

- L'espace des angles est discrétisé suivant 12 directions. Les trajectoires sont donc des segments de droite connectés par des angles de 30°. Ceci nous permet d'exploiter directement et efficacement notre carte de distance hexagonale.
- Le test de validité d'une position *(x, y, θ)* est très rapide. Le robot est modélisé par deux cercles, et donc le test de seulement deux points dans la carte de distance est nécessaire pour estimer la distance du robot aux obstacles.
- En opération normale et pour l'évitement d'obstacles proches, le robot recalcule toutes les secondes les 2 mètres devant lui et recolle cette trajectoire à la trajectoire globale (local path planning).
- L'heuristique A* appliquée limite enfin très sévèrement les branches du graphe explorées.

Par ailleurs, la carte de distance aux obstacles, fournie par le SLAM, est enrichie des osbtacles proches issus de la Kinect, et des informations en provenance du télémètre à ultrasons, très directif, qui nous fournit un point unique 10 fois par secondes (ce point nous évitant de générer des trajectoires passant par une vitre).

Le contrôle du robot lui-même est assuré par un simple contrôleur proportionnel, s'appuyant sur les différences entre la direction actuelle du robot, la direction à suivre sur la trajectoire, et la direction à prendre pour rejoindre la trajectoire avec un look-ahead de 20 centimètres. La vitesse longitudinale du robot est directement proportionnelle à la distance aux obstacles, ce qui permet d'assurer la stabilité de l'ensemble.

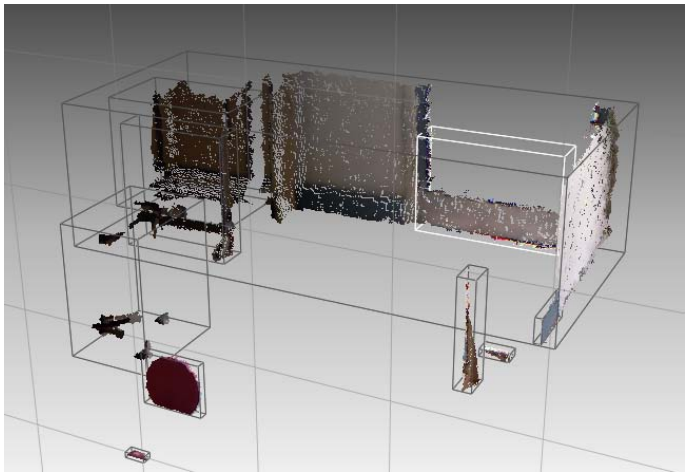

Figure 3. Segmentation du nuage de points de la Kinect en vue de la détection d'objets

CORE3DLEARNER ET CORE3DANALYZER

Notre système de reconnaissance d'objets s'appuie sur les nuages de points fournis par la Kinect. Les nuages de points colorés sont segmentés pour isoler les différents objets détectés (cf. figure 3). Ces objets sont ensuite alignés suivant leur vecteur propre principal pour constituer des modèles (*templates*) qui vont être par la suite êtres comparés à des modèles précédemment acquis. Un modèle d'objet est une image en couleur *(y, u, v)* et en profondeur de 100 x 100 avec une résolution de 1cm.

Les modèles à comparer sont sélectionnés pour chaque classe d'objets par un algorithme d'apprentissage off-line type AdaBoost (*Core3DLearner*). AdaBoost, à l'issue de la phase d'apprentissage, fournit pour chaque classe un ensemble de modèles de références associé chacun à un seuil de comparaison. Ces modèles de référence sont comparés aux modèles observés en temps réel sur le robot (*Core3DAnalyzer*), à une fréquence de 5Hz. Nous chercherons à adapter le seuil de non détection / fausse alarme pour maximiser notre score au défi CAROTTE.

CORECAROTTE2

La machine à états contrôlant la stratégie du robot s'appelle *CoreCarotte2*. Cette année, nous avons conçu une machine à états cherchant à explorer au mieux l'ensemble des éléments à identifier pour le concours : les objets à trouver, mais aussi cette année les types de sol et de mur. La stratégie consiste essentiellement à trouver un point d'observation à fort potentiel d'observation (suivant un compromis potentiel d'observation - proximité), de se diriger vers ce point, et d'opérer une rotation autour de ce point pour observer le maximum d'éléments inconnus à partir de ce point.

Un plan topologique de cellules 1m x 1m est reconstruit, permettant une identification de haut-niveau des pièces. A noter que le problème des miroirs est résolu par un "locking" : le robot, une fois sorti d'une pièce, verrouille le plan intérieur de cette dernière de manière à ce que cette partie du plan ne puisse être détruite par un mauvais mapping lié à un miroir dans une pièce adjacente. Le robot est ainsi certain de retrouver son chemin vers l'entrée.

Evidemment, CoreCarotte2 traite spécifiquement le cas de la boule rouge : une fois identifiée, le robot va chercher à la toucher. C'est un de nos objectifs pour cette nouvelle année.

CONCLUSION

Nous avons brièvement présenté dans ce papier notre nouveau robot *CoreBot M* qui sera présenté au défi CAROTTE fin juin 2011. Ce robot, toujours en cours de développement, devrait présenter des performances substantiellement améliorées par rapport au robot de l'édition 2010 : il va déjà plus vite, son contrôle est beaucoup plus précis. Si ses capacités de détection sont à l'avenant, associé à une stratégie d'exploration plus adaptée, nous devrions être en mesure de réaliser de bonnes performances. Mais comme toujours en robotique, il s'agit avant tout de soigner la fiabilité de l'ensemble qui reste très complexe. Encore donc beaucoup de travail en perspective avant d'être capable de trouver et toucher la fameuse boule rouge à coup sûr...