UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

INTÉGRATION D'UN GESTIONNAIRE DE DIALOGUES AVEC UN SYSTÈME D'AUDITION ARTIFICIELLE POUR UN ROBOT MOBILE

Mémoire de maîtrise
Spécialité : génie électrique

Maxime FRÉCHETTE

Jury : François MICHAUD (directeur)
Dominique DROUIN
Philippe MABILLEAU

Sherbrooke (Québec) Canada Décembre 2010

À ma grand-mère, Gervaise.

RÉSUMÉ

Ultimement, les humains souhaitent pouvoir interagir en dialoguant de manière naturelle avec un robot mobile. Cependant, il est très difficile d'y arriver : le language naturel comporte un ensemble de subtilités (ton, expression, formulation, etc.) qui rendent la tâche ardue, tout comme la capacité pour un robot d'entendre de manière précise et juste dans les environnements bruités et variables de la vie courante. De plus, arrimer ces interactions avec la prise de décision du robot reste une problématique ouverte : qu'est-ce que le robot doit dire, comment répondra-t-il aux requêtes exprimées par son interlocuteur, etc.

Ce projet de recherche est une première itération dans la réalisation d'un système intégré impliquant un robot mobile autonome nommé Spartacus, muni d'un système d'audition artificielle pour la localisation, le suivi et la séparation de source sonores, à lequel on couple un gestionnaire de dialogues. Jusqu'à maintenant, très peu de systèmes robotiques intègrent les fonctionnalités d'un gestionnaire de dialogues et l'audition artificielle dans le domaine de la robotique est à ses débuts. En réalisant cette intégration, nous souhaitons démontrer qu'il est faisable d'améliorer sensiblement les capacités d'interaction vocale avec un robot mobile, tout en respectant les contraintes d'exécution temps réel et de changement dynamique des contextes d'interaction du robot.

La méthodologie expérimentale consiste à intégrer ManyEars, un système d'audition artificielle développé au laboratoire IntRoLab par Jean-Marc Valin, avec l'outil CSLU (*Center for Spoken Language and Understanding*). L'outil CSLU est un système qui intègre la reconnaissance de la parole, la gestion du dialogue et la synthèse vocale, dans un environnement ouvert, flexible et convivial. L'intégration fut réalisée sur Spartacus dans le contexte de faire participer le robot à une conférence comme un participant humain. Différentes modalités d'interactions vocales ont été implémentées et validées en conditions réelles lors de la compétition AAAI 2006, ainsi que dans des conditions expérimentales en laboratoire et dans un milieu public. Les résultats obtenus démontrent que cette intégration est fonctionnelle et améliore les performances de reconnaissance vocale.

Dans cette intégration, le système d'audition artificielle et le gestionnaire de dialogues restent deux composantes distinctes, avec un couplage faible. Comme perspective de travaux futurs suite à cette première itération, la prochaine étape sera d'améliorer les performances de ManyEars en conditions bruitées (beaucoup de bruits ambiants, plusieurs interlocuteurs), et d'arriver à intégrer par un couplage fort ces deux composantes afin d'en améliorer les performances communes.

Mots-clés : Gestion du dialogue, reconnaissance vocale, synthétiseur vocal, robot mobile, interaction humain-robot.

REMERCIEMENTS

J'aimerais d'abord remercier mon directeur François Michaud pour son support tout au long de mes travaux. Sa détermination, son écoute et son support accru ont su me guider tout au long de mon séjour à l'Université de Sherbrooke.

J'aimerais également remercier Dominic Létourneau pour son soutien technique tout au long de la confection du système, du déverminage et des différents essais en laboratoire. Sa patience et ses compétences techniques ont été d'un très grand apport tout au long de mon projet.

Aussi, une attention toute particulière est accordée à tous les membres du laboratoire IntRoLab qui ont su prêter leur voix lors des différents scénarios de dialogue avec Spartacus afin de collecter des données expérimentales.

Finalement un merci rempli d'émotions est envoyé à mes parents Lise et Denis, ainsi que ma fiancée Marie qui ont su me supporter, me soutenir mais surtout m'encourager tout au long de la complétion de mes études.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LEXIQUE

Terme technique	**Définition**
Intergiciel	Logiciel intermédiaire (*middleware* en anglais)
N-gramme	Séquence de N phonèmes ou syllables pour une séquence donnée Exemple : trigramme
Plugiciel	Composante logicielle fonctionnant de pair avec un autre logiciel (*plugin* en anglais)

LISTE DES ACRONYMES

Acronyme	**Définition**
AA	*Application Adapter*
AAAI	*American Association for Artificial Intelligence*
AIML	*Artificial Intelligence Markup Language*
ASR	*Advanced Speech Recognition*
BPM	*Behavior-Producing Modules*
CMU	*Carnegie Mellon University*
CSLU	*Center for Spoken Language and Understanding*
DARPA	*Defense Advanced Research Projects Agency*
EARS	*Effective, Affordable, Reusable Speech-to-Text*
GNU GPL	*GNU General Public License*
HHM	*Hidden Markov Model*
HRI	*Human-Robot Interaction*
IFD	*Istituto Fonetica e dialettologia*
IHR	Interaction Humain-Machine
MARIE	*Mobile and Autonomous Robotics Integration Environment*
MBA	*Motivated Behavioral Architecture*
MERL	*Mitsubishi Electric Research Laboratories*
NIST	*National Institute of Standards and Technology*
NLTK	*Natural Language ToolKit*
NSF	*National Science Foundation*
ONR	*The Office of Naval Research*
UML	*Unified Modeling Language*
SSML	*Speech Synthesis Markup Language*
TCL	*Tool Command Language*
W3C	*World Wide Web Consortium*
WSJ	*Wall Street Journal*
XML	*eXtensible Markup Language*

CHAPITRE 1

INTRODUCTION

De nos jours, la recherche dans le domaine des systèmes autonomes et intelligents permet de repousser les limites des capacités des machines et d'apporter des progrès importants dans le domaine des interactions de l'humain avec les robots. Remplacer complètement l'humain par une machine intelligente afin de réaliser des tâches reste encore difficile, car la technologie actuelle est définitivement limitée par différentes contraintes physiques, mécaniques, et principalement d'intelligence. Beaucoup de travaux et de recherches doivent encore être réalisés avant de pouvoir lancer une machine dans un environnement imprévisible et dynamique où l'humain évolue quotidiennement.

Afin d'étudier et d'expérimenter des moyens pour y arriver, le robot Spartacus[1] d'IntRoLab, le laboratoire de robotique intelligente, interactive, intégrée et interdisciplinaire de l'Université de Sherbrooke, est une plateforme intelligente qui permet d'intégrer diverses capacités d'interaction humain-robot et d'intelligence. Ce robot est conçu pour participer au Challenge AAAI (*Association for the Advancement of Artificial Intelligence*), qui consiste à concevoir un robot complètement autonome et capable de participer à une conférence comme un participant humain. Le concours comprend différentes épreuves que doit réaliser le robot par lui-même. La description même du concours exprime bien les enjeux réels et l'emphase des défis à relever lors de l'événement :

> *"The goal of the Robot Challenge is to work toward the development of an interactive social robot. Toward that end, the challenge requires that the robot participate in the AAAI conference. Aspects of conference participation goals include locating the conference registration desk, registering for the conference, perform volunteer duties, and present talk (and answer questions) at a prescribed time and location. Additionally, the robot should socially interact with other conference participants. Navigational technical challenges include dynamic crowded environments, natural landmark detection, direction understanding and following, and map reading. Social interaction challenges may include natural conversation regarding the robot and the conference and personalization of conversation with recognized individuals (by name, badge, or face)."* [Rybski *et al.*, 2006]

[1] http ://www.gel.usherbrooke.ca/laborius/projects/AAAIChallenge/index.html

Le présent projet porte sur l'amélioration des capacités d'interaction vocale de Spartacus. ManyEars [Valin, 2005] est le système d'audition artificielle installée sur Spartacus. C'est le seul système actuellement disponible dans le domaine pour faire la localisation, le suivi et la séparation de sources sonores. Avec la version 2005 de Spartacus [Michaud *et al.*, 2007], le système d'audition artificielle était relié à un système simple d'interactions vocales, comportant un système de reconnaissance vocale et un ensemble de phrases pré-établies et prédéfinies à même le code source du robot. Cette façon de procéder est loin d'être optimale, car pour modifier certains aspects du dialogue, le code devait être recompilé de nouveau. Cette responsabilité est du ressort d'un système en charge d'établir de la gestion dans le dialogue. Un gestionnaire de dialogues comprend trois éléments : la reconnaissance de parole, la gestion du dialogue, et le synthétiseur vocal. En soit, la réalisation d'un gestionnaire de dialogues est un domaine de recherche important, car il implique la capacité de traiter toutes les subtilités du language naturel. Une telle problématique déborde de nos objectifs. En fait, des systèmes logiciels pour la gestion du dialogue existent, et le défi consiste alors à en intégrer un de façon appropriée sur un robot mobile opérant dans des contextes changeants et diversifiés, et effectuant les traitements en temps réel. Il existe présentement différentes plateformes robotisées qui intègrent des capacité limitées de dialogue, mais aucun robot n'intègre un système d'audition artificielle avec un gestionnaire de dialogue complet et efficace.

Pour y arriver, nous avons examiné les différents systèmes logiciels offerts pour les trois éléments impliqués dans un gestionnaire de dialogues afin de déterminer comment procéder à l'intégration de ceux-ci avec ManyEars et les autres modules décisionnels du robot (et plus spécifiquement le module de planification de tâches, qui joue un rôle important pour établir le contexte d'interaction du robot). Nous avons ensuite mis en œuvre une intégration du système ManyEars avec le gestionnaire de dialogues choisi et validé expérimentalement son utilisation dans le cadre de la compétition AAAI Challenge 2006. Cette participation fut un succès, car le robot remporta la première place dans cinq des sept catégories, dont plusieurs rattachées à l'interaction humain-robot. Cette étude pilote a permis d'établir des pistes d'amélioration et mit en lumière les difficultés de recueillir des données expérimentales uniquement sur l'intégration ManyEars-gestionnaire de dialogues dans le contexte de la compétition. Nous avons donc réalisé une étude plus spécifique des capacités de l'intégration implémentée, en laboratoire et dans un milieu public.

La structure du mémoire reflète la méthodologie expérimentale suivie pour le projet. Tout d'abord, la section 2 présente une revue des différents systèmes disponibles pour la réalisation d'un gestionnaire de dialogues. La section 3 expose le contexte dans lequel s'inscrit

le développement du gestionnaire de dialogues évolué pour le robot Spartacus. La section 4 présente l'article soumis à la revue *International Journal of Computing and Information Technology* et portant sur l'évaluation des performances de l'intégration de ManyEars avec le gestionnaire de dialogues utilisé sur Spartacus, démontrant la faisabilité ainsi que l'amélioration qu'apporte un système d'audition artificielle sur la capacité d'interagir vocalement et de façon naturelle avec des personnes. En conclusion, la section 5 revient sur les objectifs fixés et identifie des pistes d'amélioration à explorer dans des recherches futures.

CHAPITRE 2

REVUE DES SYSTÈMES DE GESTIONNAIRE DE DIALOGUES

Pour le projet, deux choix s'offrent à nous : soit rassembler les trois éléments (reconnaissance de parole, gestion du dialogue, synthèse vocale) requis pour construire un gestionnaire de dialogues, soit prendre un système qui intègre déjà les trois. Des critères de sélection ont été élaborés, et l'objectif est d'idenfier la solution qui permettra d'optimiser leur satisfaction. Les prochaines sections présentent la revue réalisée ainsi que les observations faites, pour terminer en justifiant notre choix de conception réalisée pour intégrer ManyEars à un gestionnaire de dialogues.

2.1 Système de reconnaissance vocale

Les critères à optimiser pour la reconnaissance vocale sont :

- Fonctionne en temps réel et minimise la puissance de calculs.
- Maximise la performance de la reconnaissance pour de multiples usagers.
- Utilisation multi-tâches *(threads)* possible ou tout autre mécanisme qui permet de traiter simultanément différentes trames sonores recueillies par le robot.
- Permet des modifications dynamiques au contenu de la grammaire.
- Exemples fonctionnels disponibles et support technique adéquat.

Notre analyse a permis d'identifier un certain nombre de systèmes fonctionnels mais qui ont dû être rejetés :

- NUANCE/IBM ViaVoice. Le but premier de ce système est de transcrire la parole en texte. Ceci représente un aspect intéressant, car il permet de reconnaître une large banque de mots. Le système est aussi impliqué dans différents projets visant la communication entre les humains et les machines, tels que COLLAGEN [Sidner *et al.*, 2004], [Lesh *et al.*, 2004] et Mel [Sidner *et al.*, 2004]. Cependant, le logiciel doit être entraîné à la voix de l'utilisateur pour avoir un potentiel d'efficacité intéressant.

Dans notre cas, le robot doit être en mesure d'interagir avec différentes personnes qui lui sont, la plupart du temps, inconnues.

- Open Mind Speech[1], EARS[2] et Acapela Telecom[3] ont été évalués par le *National Institute of Standard and Technology* (NIST) [Fiscus *et al.*, 2005; Hatch *et al.*, 2005]. Ils ont été éliminés à cause de l'absence de support technique (principalement via des forums de discussion), la qualité globale du produit et la fréquence de lancement de nouvelles versions.

Trois systèmes ont toutefois retenu notre attention, et sont décrits dans les prochaines sous-sections.

2.1.1 Sphinx

Sphinx est un système développé à la Carnegie Mellon University (CMU) et disponible sous forme de logiciel libre. Plusieurs versions de ce logiciel existent :

Sphinx-2 Cette version est la plus rapide de toutes les versions développées par CMU, c'est-à-dire que c'est l'engin qui permet de réaliser la reconnaissance vocale et de rendre le verdict sur le résultat dans les plus courts délais [Huang *et al.*, 1993]. Elle utilise des modèles de Markov cachés (HMM) avec des fonctions de probabilités continues en sortie. Bien qu'elle ne soit pas la version la plus efficace pour la reconnaissance vocale, elle peut cependant être embarquée sur des systèmes utilisés en temps réel qui doivent rendre des décisions dans de courts laps de temps. Pour un gestionnaire de dialogues, il est évident que le système doit être en mesure de répondre très rapidement, car il faut conserver le plus possible le réalisme dans les interactions, idéalement par un débit et un rythme qui s'approche d'une conversation entre deux personnes.

Sphinx-3 Cette distribution de Sphinx est subdivisée en deux versions [Mathew *et al.*, 2003], chacune d'elle ayant des objectifs bien distincts.

- La version 3a est développée de façon à pouvoir reconnaître le plus de vocabulaire possible. Elle utilise le même procédé de Markov que Sphinx-2 et est principalement conçue pour faire du traitement sur des groupes de fichiers préenregistrés. Son utilisation en temps réel se voit donc moins appropriée, car le temps de traitement s'avère

[1] http ://freespeech.sourceforge.net/
[2] http ://projects.ldc.upenn.edu/EARS/
[3] http ://www.acapela-group.com

plus long. Par contre, son taux d'efficacité, critère de conception de cette version, est grandement supérieur à celui de Sphinx-2.

- La version 3b est un compromis intéressant entre la version temps réel (Sphinx 2) et la version efficace de Sphinx (Sphinx 3a). Effectivement, cette version permet de traiter des trames audio en temps réel avec un plus grand nombre de possibilités de reconnaissance. Ainsi, en réduisant légèrement les performances, un plus grand vocabulaire peut être reconnu par le système. De plus, il est possible d'utiliser différents plugiciels servant à modifier les algorithmes de reconnaissance vocale.

Sphinx-4 Cette version de Sphinx est dite la meilleure solution, développée par Carnegie Mellon University. Contrairement à toutes les autres versions de Sphinx écrites en C++, celle-ci est écrite en JAVA. Elle combine deux des différents modes des autres engins, c'est-à-dire qu'il est possible de faire la reconnaissance en temps réel et de faire le traitement d'un groupe de fichiers audio enregistrés sur un média quelconque. De plus, il est possible de choisir parmi une liste de plugiciels développés spécifiquement pour certains modes de reconnaissance vocale, chacun d'eux optimisé pour des situations spécifiques. De plus, une des fonctionnalités intéressantes de cette version est qu'il est possible d'exploiter des modèles acoustiques utilisés avec la version 3a de Sphinx-3 [Walker *et al.*, 2004].

Les différentes versions de Sphinx sont toujours en développement et le soutien technique est disponible via le site web et plusieurs forums de discussion. On peut y retrouver de l'aide et du support technique très rapidement. Il est aussi à noter qu'il est possible de consulter sur le même site des résultats à des tests de reconnaissance vocale, effectués sur sur une base régulière, de chacun des différents moteurs de reconnaissance vocale. Sphinx est disponible sur la plupart des système d'opérations disponibles, tel que Windows et Linux.

2.1.2 CSLU Toolkit

Le système de reconnaissance inclus dans la boîte d'outils du *Center for Spoken Language and Understanding (CSLU)* [Center for Spoken Language and Understanding, s. d.] est basé sur un réseau de neurones artificiels qui utilise les modèles de Markov cachés et un algorithme de Viterbi pour réaliser la reconnaissance[4]. Chacun des phonèmes contenus dans les mots sont extraits et servent à entraîner le réseau de neurones. Par exemple, pour la langue anglaise, 545 différents phonèmes qui constituent les mots sont utilisés.

[4]http ://www.cslu.ogi.edu/toolkit/old/old/documentation/recog/recog.html

Le réseau contient donc 546 nœuds, un pour chacun des phonèmes à reconnaître et une sortie supplémentaire pour le bruit et les hésitations qui pourraient s'ajouter aux mots lorsqu'une personne parle. Cette technique met en évidence sa polyvalence et la rend encore plus intéressante, car le système peut arriver à reconnaître de nouveaux mots. À titre d'exemple, il est possible de citer des noms propres, des acronymes ainsi que des mots dans d'autres langues similaires, telle que le français.

La rapidité de l'algorithme de Viterbi permet de tenir efficacement une discussion complexe et ce, dans différents contextes de dialogues. Il est aussi important de noter qu'il est possible d'utiliser des grammaires chargées dynamiquement à l'aide de listes et qui peuvent être construites au moment même de la reconnaissance vocale. Ceci représente deux avantages majeurs pour le contexte d'une plateforme robotique. D'abord, notons qu'aucune compilation de grammaire n'est nécessaire avant le démarrage du système. Ensuite, il est possible de modifier dynamiquement la grammaire contenant les mots à reconnaître, même lorsque l'exécution du système est en cours. Ceci est aussi une caractéristique intéressante, car le système peut s'adapter à de nouvelles conditions tout au long de son exécution sans même avoir besoin de redémarrer le système. On peut passer d'un contexte de dialogues à un autre sans aucune difficulté ou conditions d'apprentissage supplémentaires. De plus, le système contient deux techniques permettant de retirer les bruits et les hésitations lors de la reconnaissance de parole afin de permettre au système de mieux performer.

Des forums de discussion et de mise-à-jour sont toujours actifs et le site web offre de très bons tutoriels afin de bien comprendre les principes de la boîte à outils du CSLU. Il est à noter que le CSLU fonctionne seulement sous Windows, mais qu'il peut être émulé sous Linux grâce au système WINE [WineHQ support group, s. d.], si nécessaire.

2.1.3 SONIC

SONIC est une boîte à outils qui permet la recherche et le développement de nouveaux algorithmes pour la reconnaissance vocale continue [Pellom et Hacioglu, 2003]. Depuis mars 2001, SONIC a été employé en tant que banc d'essais afin d'intégrer de nouvelles idées ainsi que pour les activités de support de recherches qui incluent la reconnaissance de la parole. SONIC fut utilisé dans des projets temps réels dont les principaux sont le DARPA Communicator [Walker *et al.*, 2000], un système de réservation de voyages ; le DARPA/NRL SPINE, [Pellom et Hacioglu, 2003], un système de reconnaissance en

milieu bruité pour le domaine militaire ; le Switchboard[5], un système de dialogues pour la téléphonie, ainsi que plusieurs autres[6].

SONIC se démarque des autres systèmes de reconnaissance vocale par ses algorithmes de traitement des trames audio. Comme certains des systèmes déjà présentés, il contient des modèles acoustiques et fonctions de modelage tels les modèles de Markov Cachés (HMM) basés sur des densités de probabilités, mais celui-ci se distingue par une recherche de solutions en deux phases. D'abord, SONIC traverse avec un algorithme de Viterbi, un arbre lexical des phonèmes de façon synchronisée dans le temps, de la même façon que le fait CSLU [McTear, 1999]. Des modèles acoustiques de language de différents types tels que les mots croisés, les trigrammes ou les quadrigrammes sont ensuite appliqués. Par la suite, une seconde passe est effectuée pour transposer le résultat sur un graphe de mots traité par un algorithme de recherche A* ou une machine à états finis. Il est facile d'interfacer SONIC avec le *CU Communicator* (aussi connu sous le nom de *Galaxy Communicator*) [Hacioglu et Pellom, 2003], une suite logicielle qui permet d'étendre les dialogues facilement, car les deux logiciels sont disponibles de façon modulaire. De plus, on constate qu'il est facile de combiner SONIC avec d'autres logiciels [Scheutz *et al.*, 2006].

Finalement, une autre caractéristique intéressante est que SONIC fonctionne sur différents systèmes d'opération, tel Microsoft Windows®, Linux, Mac OS X et Sun Solaris.

2.2 Gestion du dialogue

Les critères à optimiser pour la gestion du dialogue sont :

- Moteur de dialogues flexible et fiable.
- Code source disponible, facile à modifier et à comprendre.
- Langage de programmation connu ou facile à apprendre.
- Système utilisant des contextes ou des états de dialogues.
- Gestion des erreurs dans le dialogue possible.
- Différents modes d'interactions possible pour le robot.

Il existe un bon nombre de systèmes de gestion du dialogue. Par exemple, *COLLAGEN* (pour *COLLaborative AGENt*) est une architecture permettant de construire des systèmes

[5]http ://www.ldc.upenn.edu/Catalog/readme_files/switchboard.readme.html
[6]http ://www.bltek.com/virtual-teacher-side-menu/sonic.html

de gestion du dialogue. Elle inclut un modèle de tâches qui consiste en une série de buts de haut-niveau caractérisés par une librairie de recettes qui décompose chacun des buts en sous-buts de façon récursive. Ceux-ci sont placés dans une pile et sont complétés lorsqu'une reconnaissance vocale est réalisée, qu'une action directe est reçue ou qu'une nouvelle recette est construite. En combinant *COLLAGEN* avec le SPUD (*Sentence Planning Using Description*), il est possible de générer des discours en language naturel de façon à atteindre une flexibilité et une sensibilité que les humains recherchent lors d'une discussion avec une autre personnes (et par conséquent, avec une machine) [Lesh *et al.*, 2004]. Cet algorithme permet de générer des phrases complètes qui sont syntaxiquement correctes. Par exemple, au début de l'interaction, le système peut décider automatiquement de générer une transition où la sémantique sera la demande d'un paramètre inconnu dans une des recettes. Il est donc possible pour le système de demander le nom de l'utilisateur avec lequel il interagit, figurant que dans ce cas-ci c'est ce paramètre qui est indéfini. Par contre, il semble que le système soit quelque peu simpliste pour l'utilisation recherchée avec Spartacus. Effectivement, même si la reconnaissance vocale est indépendantes des caractéristiques vocales des usagers [DeVault *et al.*, 2004], il n'est question que de discours et de reconnaissances vocales très restreints. Dans notre cas, nous voulons que Spartacus soit en mesure d'écouter, mais surtout d'interpréter tout ce qui lui est dit dans différents contextes et qu'idéalement, son vocabulaire ne soit pas limité à un certain nombre de mots préalablement déterminés.

Les quatres possibilités valides qui s'offrent à nous sont décrites dans les prochaine sous-sections.

2.2.1 Gestion du dialogue dans le projet MARIE

En 2005, nous avons réalisé un module de gestion du dialogue avec un groupe d'étudiants travaillant au IntRoLab sur le projet MARIE [Caron *et al.*, 2005; Cote *et al.*, 2006a].

Ce module permet l'ajout de dialogues facilement et de modifier dynamiquement la tournure que peut prendre différents scénarios de discussion, d'où sa flexibilité. Le système permet de cumuler des informations simplistes telles les noms des gens, leur âge, leur sport préféré, etc.

Python est le langage de programmation utilisé dans ce module. Il contient des librairies pour le traitement des fichiers AIML[7]. Également, il inclut un logiciel permettant de traiter le langage naturel, NLTK [Bird et Loper, 2004], un projet à l'avant-plan dans le domaine

[7] http ://www.alicebot.org/aiml.html

du traitement naturel du langage, développé et utilisé dans plusieurs grandes universités américaines pour l'enseignement. Ce langage est entièrement accessible puisqu'il s'agit d'un logiciel libre et portable. Aussi, les librairies graphiques QT sont accessibles depuis ce langage. Le système de gestion du dialogue est modulaire et permet à plusieurs techniques de cohabiter. Comme le système interagit avec plusieurs autres composants tels le système de reconnaissance de la parole, l'espace de travail du robot *(Workspace)*, l'interface graphique, l'interface en mode texte, le synthétiseur de parole, etc., chacune de ces interfaces s'exécute dans une file séparée (*thread*). Pour propager les événements dans l'ensemble du système, les différentes interfaces utilisent un mécanisme d'abonnement et de publication d'événements. Une série de classes permettent de représenter les événements qui peuvent survenir dans le système.

L'AIML (*Artificial Intelligence Markup Language*) est un dérivé du XML utilisé par le gestionnaire de dialogues afin de générer une réponse à une phrase. Son but est d'associer une entrée avec une sortie conséquente. Il est possible d'utiliser des fichiers déjà disponibles sur le site de l'AIML, contenant plus de 41000 catégories de base de connaissances. Dans le module de gestion du dialogue, des étiquettes sont associées à chacun des mots pour éviter d'obtenir des structures de phrases incorrectes. Un corpus est une liste d'étiquettes qui catégorise tous les mots. Dans ce cas ci, le corpus du *Wall Street Journal* est utilisé. Dans cet immense dictionnaire, chacun des mots est associé à des attributs caractérisés par différentes probabilités. Ainsi, lorsqu'un mot possède plus d'un type d'étiquettes, le type avec la plus haute probabilité est utilisé. Avec une telle stratégie, une première itération est faite avec la reconnaissance étiquetée. Si aucune association n'est concluante, la même opération est répétée, mais seulement avec les mots non-étiquetés en entrée. Il est à noter que tous ces fichiers n'utilisent pas d'étiquettes et peuvent être interchangés dynamiquement selon les critères de personnalité désirés pour les dialogues.

L'utilisation de ce module permettrait de donner une grande flexibilité au gestionnaire de dialogues de Spartacus. Par contre, il doit être combiné à un système de reconnaissance vocale d'une grande efficacité permettant de reconnaître une grande liste de mots, et offrir une bonne flexibilité afin de profiter de toutes les fonctionalités disponibles.

2.2.2 Gestion du dialogue avec CSLU

Le CSLU Toolkit [McTear, 1999] est un gestionnaire de dialogues très complet comprenant une interface graphique qui facilite de façon très significative la mise en place d'un système de gestion du dialogue. Le système comprend des modules représentés par des états qui caractérisent des sous-ensembles de dialogues. La figure 2.1 monte un scénario de dialogues

simple. Une liste de différentes questions est associée avec des réponses connues. À chaque itération, la réponse correspondant à la question posée est sélectionnée dans une liste préalablement définie par l'usager et est ajoutée au système de reconnaissance vocale de façon à établir le bon vocabulaire à reconnaître pour cette itération. C'est avec cette technique qu'il est possible d'optimiser l'efficacité du système de reconnaissance et de faire varier la grammaire à reconnaître en ajoutant différents mots de façon dynamique.

Ainsi, le développement du scénario de dialogues est très facile et se complète en quelques instants seulement. La barre d'outils de gauche comprend la plupart des outils nécessaires à l'élaboration des scénarios de dialogues et ce, de façon très intuitive même par un utilisateur débutant [Cole *et al.*, 1999a].

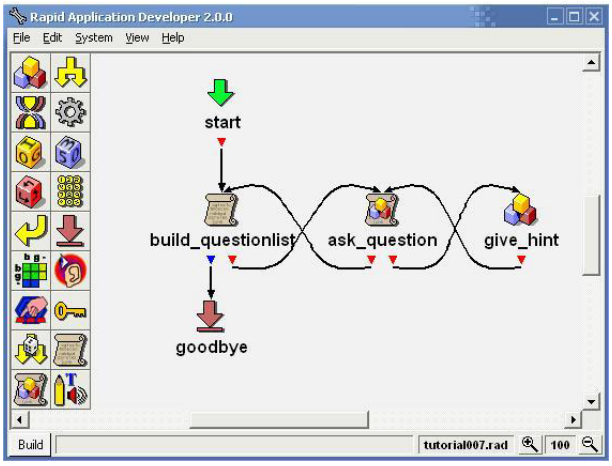

Figure 2.1 Exemple de scénario de dialogues développé avec le CSLU Toolkit

Le CSLU Toolkit est développé en TCL *(Tool Command Language)*, un langage de programmation à la fois très puissant et flexible tout en restant simple d'utilisation. Le TCL est un langage de script dérivé du C++, c'est-à-dire que son moteur utilise ce dernier pour l'exécution des commandes. Le TCL hérite donc de la rapidité et de l'efficacité du C++. De plus, le TCL comprend des outils graphiques qui facilitent le développement de fenêtres et d'interfaces graphiques.

Il est aussi important de mentionner que la boîte d'outils du CSLU permet la modification et l'ajout de commandes très facilement. Effectivement, le code source du logiciel est disponible, mais il est également possible d'ajouter des fonctionnalités à même les états du système. La figure 2.2 montre un exemple de la fenêtre des propriétés d'un des blocs de dialogues. Elle contient différents onglets où il est possible d'ajouter directement des commandes écrites en TCL, soit à l'entrée du block (*On Enter*) ou lorsque le système

sort du même état (*On Exit*). Dans cet exemple, différents fichiers contenant d'autres instructions sont chargés en mémoire.

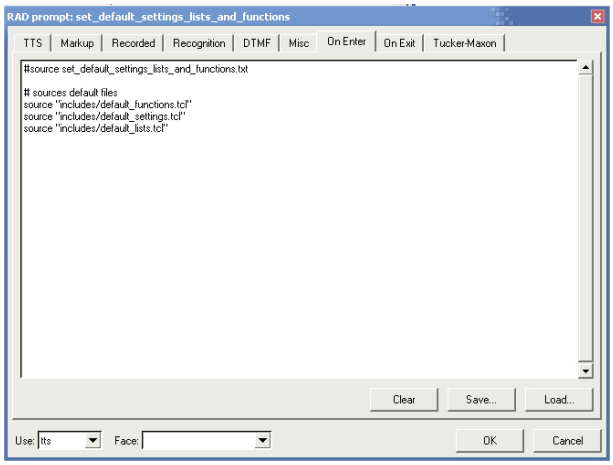

Figure 2.2 Propriétés d'un des états de dialogues du CSLU Toolkit

Chacun des états permet de spécifier des mots-clés à reconnaître, spécifiques à l'état dans lequel le système se trouve. De cette façon, il est possible d'optimiser la reconnaissance, car plus il y a de mots à reconnaître dans une grammaire, plus une reconnaissance exacte et précise est difficile. Ce problème est la limitation majeure du système de reconnaissance de Spartacus, car la grammaire contient tous les mots utilisés dans chacun des états du système. C'est pourquoi il est difficile de reconnaître les bons mots au bon moment, et ceci dans le bon contexte. De plus, dans la boîte à outils du CSLU, différents blocs permettent la reconnaissance bloquante de mots-clés [McTear, 1999].

Il est très facile de réaliser la génération de réponses avec le CSLU. Effectivement, il permet de définir des états pour chacune des parties de dialogues, et il est possible de créer différentes listes de réponses associées à chacun de ceux-ci et les utiliser par la suite. Ceci permet donc de varier le dialogue, en ajoutant une série de réponses prises aléatoirement lors de chacune des itérations dans ce même état.

Une autre caractéristique notable de ce système est qu'il inclut un mécanisme de réparation de dialogues. Effectivement, lorsque le système éprouve des difficultés à reconnaître les dires d'un utilisateur, un sous-système est appelé et demande soit de répéter ou de clarifier ce qui vient d'être dit, soit de rendre l'algorithme de reconnaissance moins stricte et plus permissif. Il en résulte une gestion d'erreur efficace, qui est aussi modifiable à même l'interface graphique du CSLU. Le système peut donc demander de confirmer ou d'infirmer la reconnaissance vocale auprès de l'utilisateur. De plus, ce système de réparation permet de réagir adéquatement à des situations connues, ce qui est très intéressant lorsqu'il est question d'entretenir une discussion entre un interlocuteur humain et un robot.

Selon la documentation du CSLU, les requis en puissance de calculs sont minimes : il est recommandé d'avoir minimalement un processeur cadencé à 200 MHz afin de pouvoir réaliser adéquatement la reconnaissance vocale.

La boîte d'outils du CSLU inclut un système permettant de générer des expressions faciales synchronisées avec les trames sonores qui sont envoyées vers le générateur de voix [Cole *et al.*, 1999b]. Ceci représente une caractéristique intéressante pour l'intégration à un robot autonome avec une interface faciale, afin de lui donner une personnalité et de rendre plus vivant le dialogue. Effectivement, comme le but premier de la plateforme est de faciliter les relations humain-robot, cet ajout est considérable pour les dévelopements futurs des capacités du gestionnaire de dialogues.

Enfin, le logiciel offre un soutien technique intéressant et polyvalent, car l'équipe du CSLU supporte des forums qui sont toujours actifs. Les réponses y sont affichées rapidement, soit de la part des modérateurs ou des autres usagers. Des exemples de codes sont fournis.

2.2.3 Galaxy Communicator

Galaxy Communicator est un système distribué de gestion du dialogue, fonctionnant selon les principes fondamentaux d'un réseau informatique. La figure 2.3 montre l'architecture logicielle du système. Avec son architecture distribuée, il est possible de changer le module de la reconnaissance vocale sans en affecter le reste du système. De plus, comme chacun des modules est indépendant, il est possible de les arrêter en tout temps et de les redémarrer par la suite.

Commanditée par le DARPA (*Defense Advanced Research Projects Agency*), la qualité du logiciel et sa stabilité sont excellentes. De plus, cette suite est un logiciel libre, ce qui est un élément important lorsque l'on désire accéder ou modifier des parties du logiciel [Bayer *et al.*, 2001]. Advenant le besoin de comparer différents systèmes de reconnaissance vocale ou de synthétiseurs de voix, cet outil s'avère une solution intelligente, car il est très facile d'interfacer de nouveaux modules avec l'infrastructure.

Il est aussi important de mentionner que le système possède la capacité d'acquérir des connaissances simples. Effectivement, lorsqu'il tente de remplir une tâche courante, le système retient certaines données afin de pouvoir les répéter ou des les retransmettre. Ceci est intéressant pour le système, car Spartacus doit être capable de démontrer une progression dans ses discours lors de son évolution dans le temps.

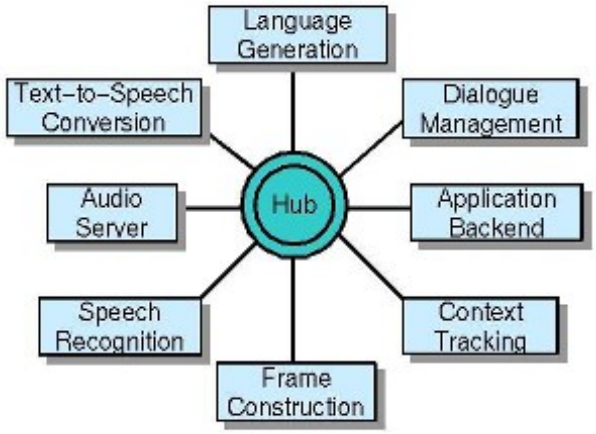

Figure 2.3 Architecture distribuée du logiciel *Galaxy Communicator*

Différentes implémentations ont été réalisées avec *Galaxy Communicator*. Une des plus intéressantes est AGENDA [Rudnicky et Wu, 1999]. Le système est toujours actif et il est possible d'interagir avec celui-ci via le téléphone. La grammaire contient environ 2500 mots, dont environ 250 sont des destinations (noms propres) aux États-unis et 500 autres ailleurs dans le monde. Afin de réaliser la reconnaissance vocale, le système utilise Sphinx-2 en temps réel et supporte l'interruption vocale (*barge-in*). Cette caractéristique est intéressante, car il est possible d'interrompre le système en tout temps afin d'accélérer les interactions.

Un autre dérivé du *Galaxy Communicator* est le module de gestion du dialogue RavenClaw[8]. Sa base est constituée du squelette d'AGENDA :

> *"The RavenClaw dialog management framework enforces a clear separation between the domain-specific and the domain-independent aspects of the dialog control logic. System developers focus exclusively on specifying the domain-specific aspects which are captured by the dialog task specification, essentially a hierarchical-plan for the interaction. In parallel, the RavenClaw dialog engine automatically ensures a rich set of conversational behaviors, such as error-handling, timing and turn-taking, etc."* [Lee *et al.*, 2010]

Comme il a été mentionné précédemment, l'architecture du *Galaxy Communicator* permet d'intégrer facilement différents composants. Effectivement, le système de reconnaissance vocale utilisé est celui de NUANCE Communications et le synthétiseur de voix est celui de la compagnie Cepstral. C'est pourquoi cette architecture renferme un potentiel intéres-

[8] http ://www.cs.cmu.edu/ dbohus/ravenclaw-olympus/

sant quant à la mise à jour du système de dialogues de Spartacus, puisqu'il est possible d'interfacer facilement différents logiciels ensembles.

2.3 Synthétiseur vocal

Les critères à optimiser pour la synthèse vocale sont :

- Permet de choisir une voix synthétique, conviviale, dynamique et expressive.
- Intonation, ponctuation et syntaxe modifiables selon les contextes de dialogues.

Parmi les solutions évaluées précédemment, NUANCE offre un générateur de trames à partir de texte. Cependant, NUANCE est un logiciel propriétaire et son code source n'est pas disponible. La compagnie Cepstral offre aussi un logiciel de synthétiseur de voix intéressant, avec une multitude de voix disponibles sur différentes plateformes telles Microsoft Windows® et Linux. Le logiciel supporte le *Speech Synthesis Markup Language (SSML)*[9], qui définit une panoplie de méthodes qui permettent de modifier la sonorité de la voix lors de sa sortie du synthétiseur de voix. Il offre aussi des utilitaires qui permettent de générer des trames sonores à partir de fichiers textes déjà présents sur un média, ou simplement en entrant des données dans l'éditeur de texte.

La boîte à outils du CSLU contient aussi un générateur de trames vocales. Effectivement, il s'agit du logiciel libre Festival [Black et Taylor, 1997], mais avec quelques ajouts intéressants. Effectivement, il est possible d'ajuster dynamiquement et facilement le débit de la voix à chaque génération de trames et de modifier le pas de variation de la voix. Plus concrètement, il est possible de transformer la voix pour la rendre plus grave ou plus aigüe, permettant par exemple de simuler différentes personnalités tels un enfant ou une femme. De plus, en combinant les deux effets, il est possible de rendre la voix plus vivante, car les variations simulent des émotions dans la voix. Une accélération dans le ton permet de simuler une personne pressée par le temps et une décélération laisse transparaître le contraire.

2.4 Sommaire et décision

Après avoir examiné les différentes solutions qui s'offrent à nous pour arriver à réaliser un gestionnaire de dialogues, la solution la plus complète semble être CSLU. Ce système intègre directement la reconnaissance de parole, la gestion du dialogue et la synthèse

[9] http ://www.w3.org/TR/speech-synthesis/

vocale, dans un environnement où il est possible de pouvoir réaliser une intégration avec le système d'audition artificielle ManyEars. C'est donc avec cet outil que nous réalisons nos travaux.

CHAPITRE 3

CONFIGURATION MATÉRIELLE ET LOGICIELLE DE SPARTACUS SANS GESTIONNAIRE DE DIALOGUES

Le présent projet de maîtrise a débuté suite aux travaux réalisés sur Spartacus pour sa participation au Challenge AAAI de 2005 [Beaudry *et al.*, 2005; Michaud *et al.*, 2005a, 2007, 2006; Valin *et al.*, 2004]. Spartacus est un robot de forme humanoïde équipé d'un laser SICK LMS200 qui lui permet de percevoir les formes des objets sur un plan en deux dimensions dans son environnement, une caméra Sony SNC-RZ30N 25X *pan-tilt-zoom*, une matrice de huit microphones placés à même la coque du robot, un écran tactile, un amplificateur sonore avec haut-parleurs, un dispositif permettant de remettre des cartes d'affaire et un panneau à affichage composé de DEL (Diode Électro-Luminescente). Il possède également un ordinateur embarqué de type Mini-ITX dans son torse, cadencé à une fréquence de 1.7 GHz. Les communications entre l'ordinateur et les périphériques se font par différents bus de données : les microcontrôleurs de bas-niveau sont reliés via un bus CAN ; la communication avec le laser se fait par un port série ; la caméra est branchée sur un réseau Ethernet 100 Mbps et le système de son utilise simplement la sortie audio [Michaud *et al.*, 2005a]. Un deuxième ordinateur portatif (Pentium M 1.8GHz) est installé dans le dos de Spartacus. C'est ce dernier qui traite les informations perçues par les micros, chacun utilisant un des ports de la carte de sons *RME Hammerfall DSP Multiface*. Les deux ordinateurs communiquent entre eux via un réseau local Ethernet ayant une vitesse de 100Mbps. Le système d'exploitation de chacun des ordinateurs est Debian GNU Linux. Les communications inter-modulaires se font principalement à l'aide de l'intergiciel MARIE[1] [Cote *et al.*, 2006a,b].

Afin de pouvoir réaliser la reconnaissance vocale avec un maximum d'efficacité, un système d'acquisition et de traitement de signaux audio a été conçu [Valin *et al.*, 2004], référencé sous l'appellation ManyEars. Ce système, composé de huit micros, est présent sur Spartacus avec le logiciel de reconnaissance vocale NUANCE [Michaud *et al.*, 2007]. Configuré avec quatre micros à l'avant et quatre à l'arrière, comme l'illustre la figure 3.1, le système permet de filtrer, d'isoler et de localiser les différentes sources sonores dans un environ-

[1] http ://marie.sourceforge.net/

nement quelconque. Plus spécifiquement, dans une ambiance de foule, il est possible de réduire le bruit ambiant et d'isoler plusieurs trames sonores simultanément, i.e., différentes personnes discutant entre elles ou non dans le même environnement que le robot [Valin *et al.*, 2004]. Cette fonctionnalité se démarque des autres systèmes d'interactions vocales, tels que les systèmes sur Grace [Simmons *et al.*, 2003], Kismet [Breazeal, 2004] et PaPeRo [Nuttin *et al.*, 2004], puisque la perception vocale ne se fait, dans ces cas, qu'avec un seul microphone.

Figure 3.1 Configuration à huit micros de Spartacus

L'architecture décisionnelle MBA, utilisée pour l'implémentation de l'intelligence de Spartacus, est stimulée par des sources motivationnelles qui examinent différentes influences affectant les intentions du robot [Beaudry *et al.*, 2005; Michaud *et al.*, 2006]. Selon l'importance de celles-ci, des recommandations sont envoyées au sélecteur de tâches qui active différents comportements se transformant en une série de commandes envoyées aux actionneurs. La figure 3.2 présente l'architecture décisionnelle utilisée sur Spartacus [Michaud *et al.*, 2005a].

Pour les interactions vocales, Spartacus possède trois éléments distincts : un système de reconnaissance vocale, un module de gestion du dialogue et un synthétiseur de voix. Le système de reconnaissance vocale est NUANCE version 8, qui comporte plusieurs limitations et lacunes, telles que :

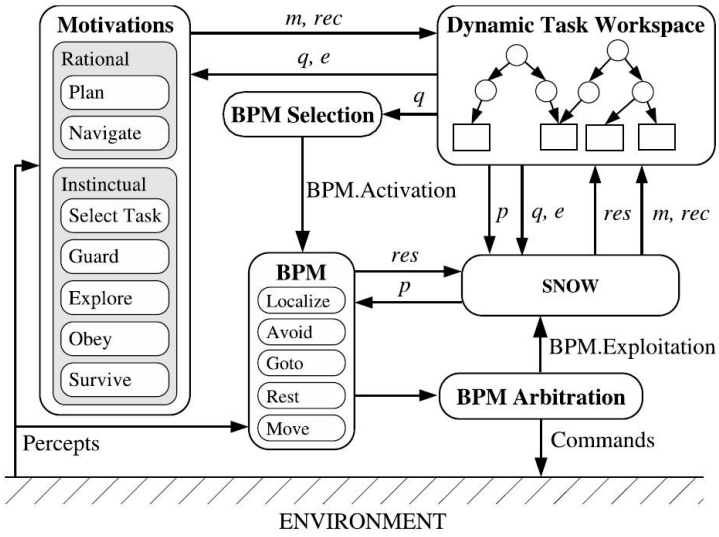

Figure 3.2 Architecture MBA de Spartacus

- Code source propriétaire non accessible ;
- Format de la grammaire spécifique à la compagnie NUANCE (et non selon les standards établis par le W3C[2]) ;
- Grammaires dynamiques présentement non-fonctionnelles ;
- Utilisation de grammaires statiques qui doivent comprendre tous les mots à reconnaître ;
- Grammaire contenant un vocabulaire très limité ;
- Logiciel non libre nécessitant une licence ;
- Reconnaissance de locuteur non fonctionnel avec la machine d'état du code source ;
- Plusieurs anicroches dans la documentation et les exemples fournis ;
- Support de la compagnie inexistant à l'heure actuelle ;
- Système de reconnaissance complexe et lourd pour l'ordinateur du robot ;
- Exemples fournis majoritairement écrits en JAVA (l'utilisation actuelle sur Spatacus est réalisée en C++).

Ce sont principalement ces caractéristiques du système de reconnaissance qui ont été évaluées quant à la décision de son remplacement. Ceci a permis de conclure que, dans un

[2]http ://www.w3.org/TR/speech-grammar/#S2.2.3

monde idéal, il serait intéressant que le moteur de reconnaissance possède les caractéristiques suivantes :

- Code source disponible sous licence GPL ;
- Grammaires statiques et dynamiques fonctionnelles et respectant les standards de la W3C ;
- Possibilité de réaliser la reconnaissance de locuteur ;
- Documentation et support disponible et efficace ;
- Système qui permet l'ajout d'extensions ;
- Noyau permettant le traitement simultané de plusieurs trames sonores ;
- Intégration facile avec l'architecture actuelle du robot.

De plus, la plupart des systèmes de reconnaissance vocale utilise une grammaire pour construire les différentes structures linguistiques à reconnaître. Pour un système simpliste, qui ne contient qu'un seul procédé de reconnaissance, il est impossible pour le système de la "comprendre", de la reconnaître, car les mots à identifier lors du processus n'apparaîssent pas dans la banque de mots constituant la grammaire. La version de NUANCE utilisée sur le robot ne donne aucun autre moyen d'apprentissage que les grammaires statiques, qui comprennent chacun des mots à reconnaître. Ceci limite donc beaucoup la possiblité de reconnaître des phrases complexes. À chaque fois que l'on veut ajouter des formes *(patterns)* à la grammaire, on doit la recompiler et redémarrer le système audio. Cette façon de procéder est loin d'être optimale pour un système qui doit fonctionner en temps réel, mais surtout de façon intelligente et autonome.

Le principal rôle du module de gestion du dialogue est d'activer les différents modes de reconnaissance vocale selon le contexte des interactions vocales, et de s'occuper des multiples envois au synthétiseur de voix. Il s'agit donc d'une façon logique de centraliser les prises de décisions liées au dialogue dans un même bloc. Le module de gestion du dialogue peut aussi être vu comme une machine contenant différents états qui évoluent dans le temps. Dans sa version 2005, aucune gestion évoluée du dialogue n'est réalisée sur Spartacus. Un *Application Adapter*[3] dans MARIE joue le rôle de module de gestion du dialogue, et relie le système de reconnaissance vocale et le synthétiseur de voix. De plus, le *Application Adapter* utilisé pour la gestion du dialogue vient à l'encontre d'un des principes fondamentaux de MARIE, car pour modifier le contexte nécessaire à la reconnaissance

[3]http ://marie.sourceforge.net/Project.html#AA

vocale, un fichier texte est utilisé. Cette façon de procéder est loin d'être optimale et doit donc être modifiée afin de centraliser le contrôle et le contexte de la reconnaissance au module de gestion du dialogue. Il n'existe alors aucune logique flexible entre les deux modules qui permet d'ajuster le contenu du dialogue de façon simple et rapide. Il est aussi impossible de cumuler des informations reçues des personnes qui interagissent avec Spartacus, car aucun mécanisme permettant l'acquisition de données pertinentes n'est intégré.

Enfin, Festival[4], un logiciel libre, est utilisé sur le robot et permet de prendre une suite de mots sous format texte et de les transformer en une trame sonore. C'est de cette façon que l'on peut simuler une voix pour le robot. Festival est simple d'utilisation et fonctionne très bien. Cependant, la voix présentement utilisée est loin de favoriser les interactions naturelles. Effectivement, le robot s'exprime comme un robot, c'est-à-dire que son expression vocale est saccadée et quelque peu inappropriée lorsqu'il est question d'inciter les gens à dialoguer avec lui. Par contre, il est possible de modifier la tonalité, le débit et les voix avec Festival de façon à rendre le logiciel plus vivant lors des interactions.

C'est à partir de ces considérations que le développement du gestionnaire de dialogues évolués, interfacé avec le système d'audition artificielle, fut mis en œuvre. Une nouvelle implémentation de l'architecture décisionnelle de Spartacus fut réalisée afin de participer à la compétition AAAI Challenge 2006. L'annexe A présente l'article qui décrit la configuration de Spartacus présentée lors du AAAI Challenge 2006. Il couvre plus large que le travail pertinent pour ce mémoire et explique en détails le robot, ses capacités perceptuelles (dont le système d'audition artificielle), ses modules décisionnels et les scénarios d'interaction implémentés, auxquels s'intègre le gestionnaire de dialogues. Les informations présentées permettent également de mettre en évidence le contexte dans lequel l'intégration du gestionnaire de dialogues fut réalisé, avec ManyEars (identifié sous l'appellation *Sound Source Localization, Tracking and Separation*, ou *SSLTS*), l'architecture décisionnelle et le planificateur de tâches du robot, afin de situer les choix de conception réalisée. Il est également possible d'y lire les conclusions tirées suite aux résultats obtenus lors de différentes épreuves. Les contributions directes du travail réalisé pour ce mémoire sont présentées aux sections A.4 (figure A.2) et A.4.1 (la partie intitulée *Interaction modalities*).

[4]http ://www.cstr.ed.ac.uk/projects/festival/

CHAPITRE 4

INTÉGRATION D'UN GESTIONNAIRE DE DIALOGUES AVEC UN SYSTÈME D'AUDITION ARTIFICIELLE POUR UN ROBOT MOBILE

Auteurs et affiliations :

- M. Fréchette : Étudiant à la maîtrise, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.
- D. Létourneau : Professionel de recherche, Laboratoire de robotique intelligente, interactive, intégrée et interdisciplinaire (IntRoLab), Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.
- F. Michaud : Professeur, Laboratoire de robotique intelligente, interactive, intégrée et interdisciplinaire (IntRoLab), Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

État de l'acceptation : Soumis pour évaluation le 11 novembre 2010

Journal : Journal of Computing and Information Technology (JCIT)

Titre français : Intégration d'un gestionnaire de dialogues avec un système d'audition artificielle pour un robot mobile

Contribution au document : L'article focalise sur l'intégration du système d'audition artificielle avec un gestionnaire de dialogues, et l'analyse des performances résultantes.

Résumé français : Afin de démontrer l'influence de la localisation, du suivi et de la séparation de sources sonores pour la reconnaissance vocale et la gestion de dialogues pour un robot mobile, cet article présente une étude de cas comprenant l'intégration des ManyEars, un système de localisation, de suivi et de séparation de sources sonores, avec le CSLU Toolkit, un gestionnaire de dialogues. Des essais ont été réalisés dans des environnements bruités et difficiles, en laboratoire et dans une cafétéria. Les résultats indiquent que le pré-traitement des trames audio réalisé par ManyEars améliore les performances de

la reconnaissance vocale et du système de gestion de dialogues, et démontrent que cette intégration rend possible pour un robot d'interagir avec des interlocuteurs dans une grande variété de contextes et de conditions.

4.1 Abstract

To demonstrate the influence of an artificial audition system on speech recognition and dialogue management for a robot, this paper presents a case study involving soft coupling of ManyEars, a sound source localization, tracking and separation system, with CSLU Dialogue Management system. Trials were conducted in laboratory and public places. Results indicate that preprocessing of the audio signals by ManyEars improves speech recognition and dialogue management of the system, demonstrating the feasibility and the added flexibility allowing a robot to interact vocally with humans in a wide variety of contexts.

4.2 Keywords

Dialogue management, sound source localization and separation, natural language processing.

4.3 Introduction

Giving robots the ability to process natural language comes with great challenges, as they have to operate in changing and diverse conditions. Natural language systems usually process audio streams recorded from one microphone using three main components : 1) a speech recognition module, (e.g., Sphinx [Huang *et al.*, 1993], NUANCE) ; 2) a dialogue manager (e.g., COLLAGEN [DeVault *et al.*, 2004], MIT's Galaxy Communicator [Bayer *et al.*, 2001]) ; 3) a text-to-speech synthetiser (e.g., Festival [Black et Taylor, 1997], Gnuspeech). These systems usually assume that speech is acquired from a microphone located close to the interlocutor (usually attached on a headset) in order to generate clear audio streams. However, this assumption is not valid for a mobile robot operating in open settings, interacting with multiple people and in different contexts.

Recently, a sound source localization, tracking and separation system called ManyEars [Badali *et al.*, 2009; Briere *et al.*, 2008; Valin *et al.*, 2007a,b] has been released [IntRoLab, 2010]. This system is also used by Kyoto University's HARK system [Nakadai *et al.*, 2008, 2010]. It consists of an array of eight microphones placed on the robot's body. The localization and tracking algorithm is based on a frequency-domain implementation of a steered beamformer along with a particle filter-based tracking algorithm. Results show that a mobile robot can localize and track in real-time up to four moving sources of different types over a range of 7 meters. Sound source separation is accomplished with

a real-time implementation of geometric source separation (GSS) and a postfilter that gives a further reduction of interference from other sources. Compared to using one microphone, ManyEars has shown to greatly improve word recognition of simultaneous speech in controlled conditions (using recordings and three loudspeakers) from a 10% recognition rate, to 25% (with a 10° angle between the center speaker and the side loudspeaker, which is more difficult to separate because of the proximity of the sources) and up to 72% (with a 90° angle between the center speaker and the side loudspeaker) recognition rate [Valin *et al.*, 2007b].

Up to now, ManyEars has been used mostly for localizing sound sources and as a preprocessing module for speech recognition of words, and has not yet been integrated with a natural language processing system. To evaluate if ManyEars can improve performances for speech recognition and dialogue management of a robot operating in natural settings, we decided to conduct a case study integrating ManyEars to CSLU (Center for Spoken Language Understanding) Toolkit [Cole *et al.*, 1999a,b], a dialogue management system. We chose CSLU because it is complete system and it offers interesting features, such as :

- Rapid Application Developer (RAD) [McTear, 1999], a graphical tool to easily create dialogue scenarios ;

- more accurate and more realistic voice interactions by easily and quickly change voices, pitches and rates [Cole *et al.*, 1999a] ;

- tokens that can be added before and after valid grammar strings, to add flexibility in recognizable speech patterns ;

- the ability to dynamically change grammars used for speech recognition. This is an important feature because it allows the system to add or remove words in the system's lexicon according to the interaction context (which depends on the robot's current task), optimizing processing time and making it possible to adapt to the robot's intention.

- a dialogue repair tool used to repair dialogues when the result of speech recognition does not meet a predefined threshold of efficiency. In our case, the process consists of conducting a second iteration of speech recognition on the audio stream with a more flexible grammar, by adding garbage collectors around the grammar itself and by dynamically changing the out of vocabulary rejection and word spotting medians, allowing the recognizer to be more permissive during its second pass [McTear, 1999].

- an optimized version of the University of Edinburgh's Festival [Black et Taylor, 1997] package for text-to-speech synthesis ;

This paper presents how ManyEars can be used as a preprocessing module for CSLU on a mobile robot, and analyze the results obtained from having people interact vocally using complete sentences (and not just words) with a robot, in laboratory conditions and in a cafeteria. The paper is organized as follows. Section 4.4 presents the experimental setup and implementation used in our evaluation. Section 4.5 follows with a description of the trials and results, and Section 4.6 concludes with a discussion on the observed performances and future work.

4.4 Integration of ManyEars and CSLU Toolkit

Figure 4.1 illustrates the architecture of the natural language processing systems implemented using ManyEars and CSLU. The system runs on three computers. Computer 1 is equipped with a multi-input sound card and is dedicated to ManyEars' audio processing algorithm and generate the audio streams to process. Computer 2 runs CSLU's Toolkit for dialogue management. Computer 3 hosts the decision-making processes of the robot, described in details in [Michaud *et al.*, 2009]. Audio streams processed by ManyEars are sent to the CSLU Toolkit for recognition, evaluation and decision. When required, vocal responses are synthesized to audio streams by CSLU Toolkit and are sent to the audio server for playback on the robot's loudspeakers.

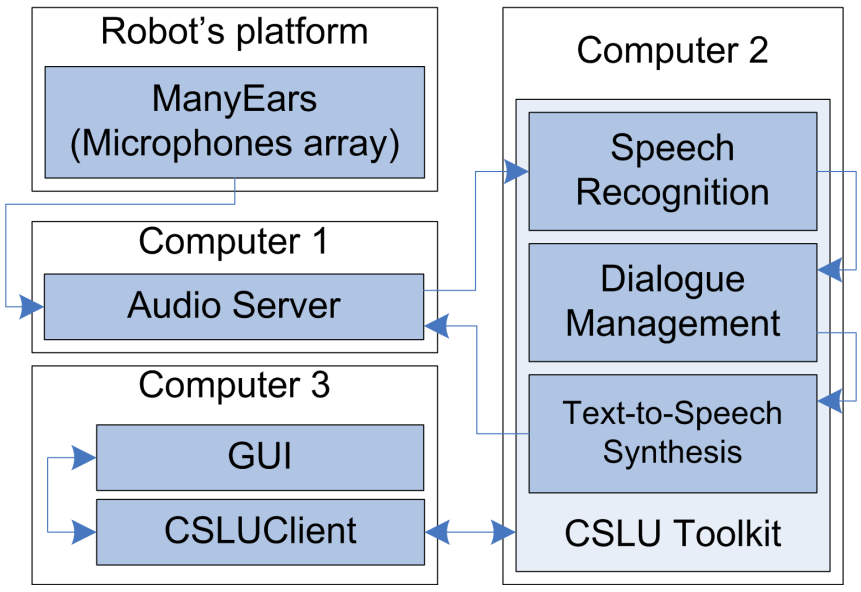

Figure 4.1 Natural language processing architecture using ManyEars.

Figure 4.2 shows the Spartacus robot used in the trials. Our human-robot interaction framework involves having interlocutors talk to the robot to respond to questions or to navigate vocally through graphical user interface (GUI) windows displayed on the robot's touch screen. Components of GUI windows (buttons, fields, widgets, etc.) are added or removed to the grammar (formatted according to a modified version of the W3C Speech Recognition Grammar Specification) as they are made available by the robot to the interlocutors. Dynamic changes to the grammar is done through the CSLUClient module which communicates with the CSLU Toolkit module using a network socket. When a recognition is performed successfully by CSLU's Speech Recognition module, or when a decision is taken by the Dialogue Management module, a message is sent back to the CSLUClient module, which in turn interprets the requests and executes the related tasks on the robot. Figure 4.3 shows an example of a GUI window as well as a text-to-speech on-screen display.

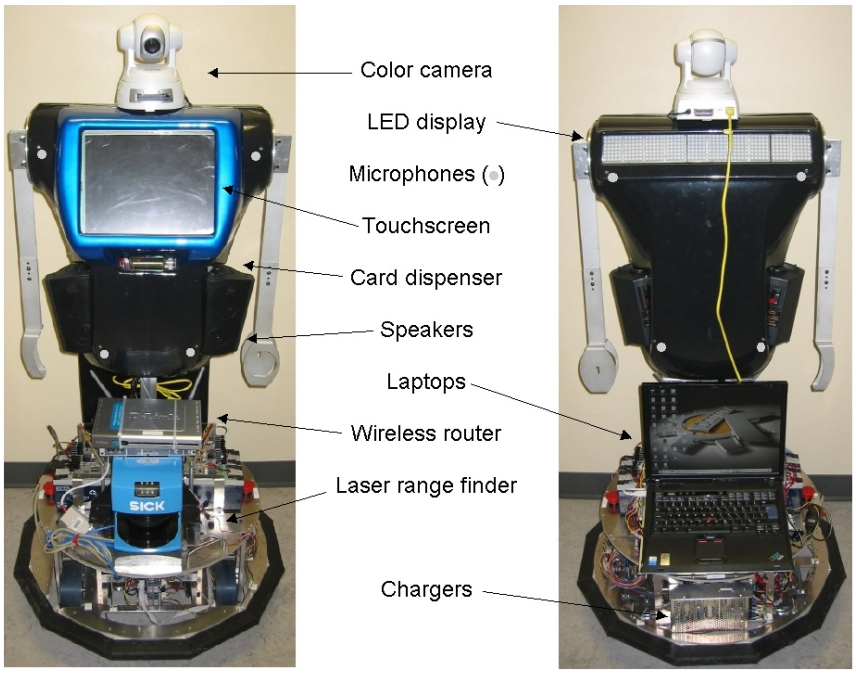

Figure 4.2 Spartacus platform used in our trials.

Figure 4.4 illustrates the flow diagram of the Dialogue Management module. At startup, the system initializes the system's parameters. It then enters an infinite loop, through the state *check_system_status*, which evaluates five conditions :

- *set_values*. This element makes it possible to dynamically load new grammars, as determined by the robot's decision-making processes and communicated through CSLUClient. Grammars processing and pronunciation extraction for speech processing require important processing time, and the robot has to run in real-time. Therefore, when a new grammar must be loaded into the system, pronunciations are extrac-

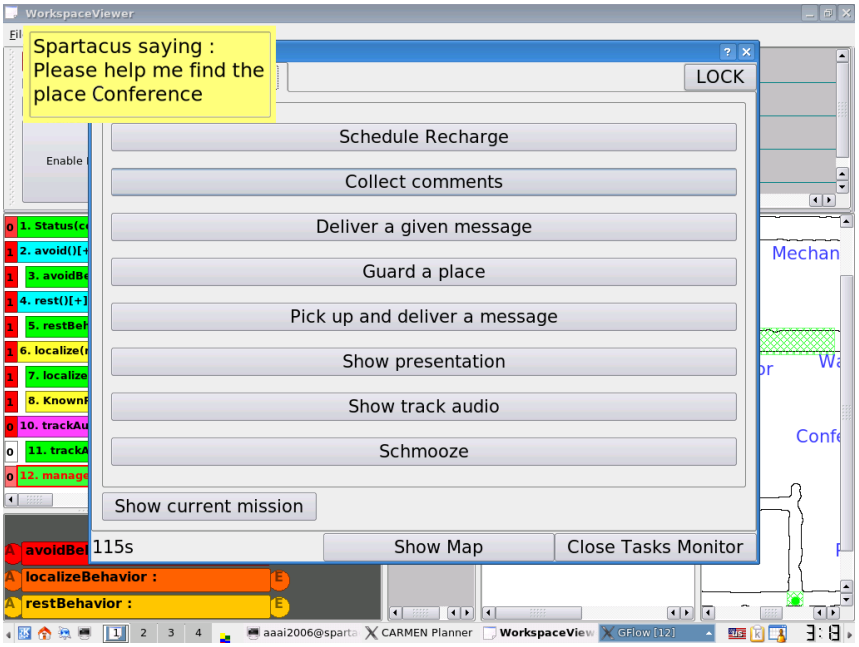

Figure 4.3 Robot's tasks manager and text-to-speech on-screen display.

ted, grammar attributes are set, text-to-speech strings and state change requests are loaded in the associated state blocks (*recog_ speech*, *perform_ TTS*, etc.). A confirmation is sent back to CSLUClient upon receiving valid data. The grammar is copied in memory and saved in a file when the system is stopped. When the CSLU Toolkit is later restarted, it reopens the file, reads all saved values and stores them in memory. This avoids having to process grammars every time the dialogue context changes, decreasing execution time from a few seconds to milliseconds.

- *perform_ TTS*. This blocks performs text-to-speech (TTS) synthesis by sending the audio streams to the audio server for playback, and to CSLUClient to display on the GUI. This occurs when the robot's state changes and before speech recognition can continue, because we want the user to be aware of the interaction context with the robot. For instance, if the user select (either vocally or by touch) a button on the screen, the robot would say out loud its new task.

- *get_ state*. The robot has different pre-programmed interaction modes : *Trivias*, *jukebox*, *fortune_ cookie*, *Schmooze* and *entertain*. These modes are activated by the interlocutor through vocal or touch interaction with the GUI windows.

- *recog_ speech*. When new audio data is received from the audio server, speech recognition is performed using the prevailing grammar. Valid recognitions are sent to CSLUClient via *send_ results* and a corresponding action is taken by the robot. Otherwise,

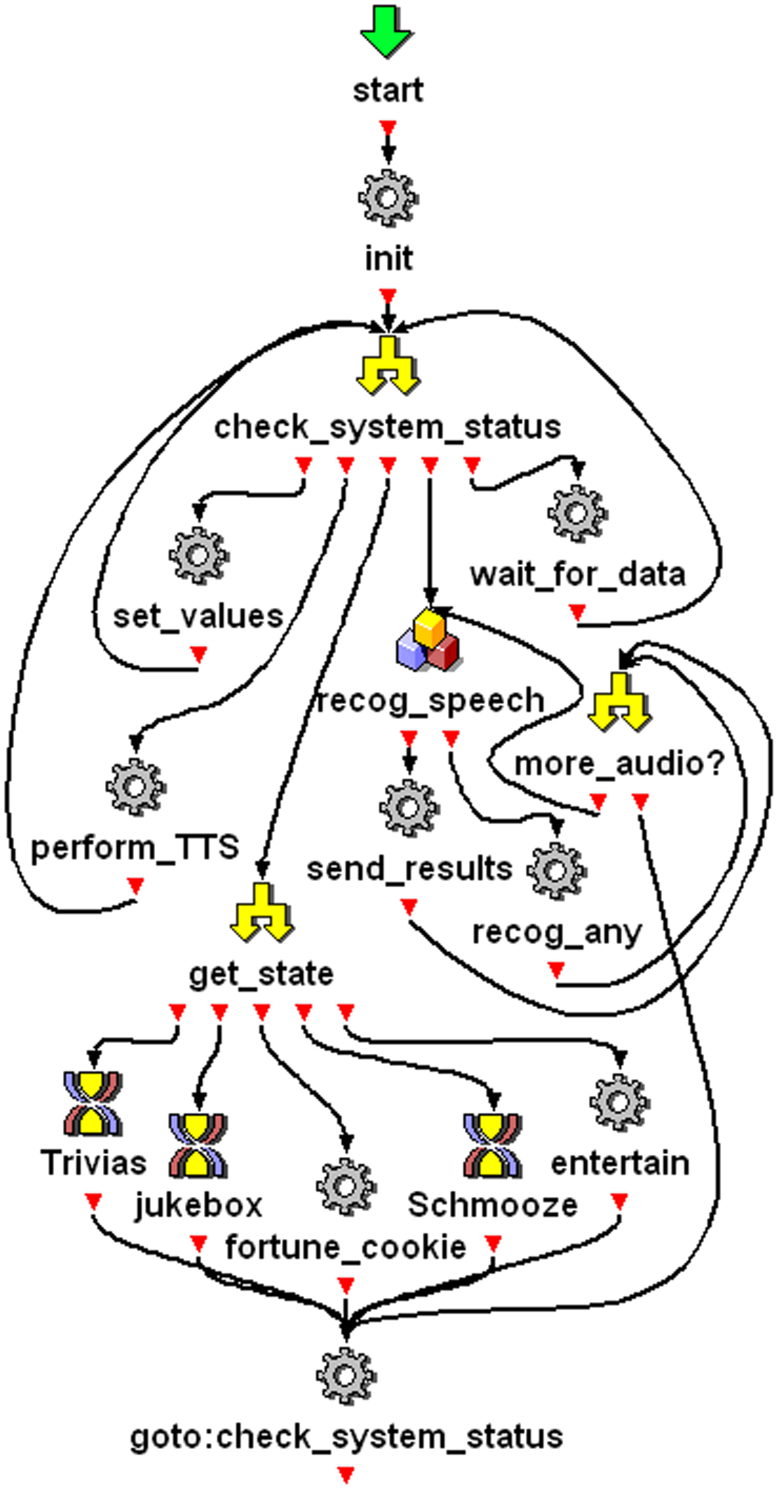

Figure 4.4 Flow diagram of the Dialogue Management module.

CSLU dialogue repair is applied (*recog_ any*). If repair fails, depending on the repair state, the system checks if there is additional audio streams (*more_ audio ?*) and continues to perform speech recognition.

- *wait_ for_ data*. When there is no configuration requests or audio streams to process, the system puts itself to sleep to minimize processing power.

4.5 Evaluation and Results

Tests were conducted using a convenience sample of 12 healthy students in the Faculty of Engineering of the Université de Sherbrooke. Participants were asked to navigate vocally through the GUI windows, and to provide answers to requests made by the robot. For both type of interactions, participants were asked to formulate complete sentences starting with the robot's name, and then make their request or respond just like they would normally do. As an example, in the specific context shown in Figure 4.3, it would have been possible for the user to select the "Schedule Recharge" button by saying "Spartacus, press the Schedule Recharge button please", or to close the Tasks Monitor window by stating "Spartacus, push the Close Tasks Monitor button". In the same manner, answering the question "In which city were the 2010 olympic games ?" asked in the Trivia mode would be done using a complete sentence like "Spartacus, the answer is Vancouver"). Audio streams shorter than 1 sec were discarded by default because they had to be longer to form complete sentences.

Trials were conducted in two environments : in the lab (LAB) and in a cafeteria (CAF). Comparing the two, the lab environment had less background noise compared to the cafeteria, the latter being much more challenging for ManyEars to separate sound sources. Both environments were not controlled, i.e., our system was used in the natural conditions of these environments, with multiple people talking and noise coming from different sources (chairs, door closing, laughter, etc.). More trials were conducted in the lab environment because this is where the added benefit of using ManyEars with a dialogue management system can be best evaluated. In the cafeteria, performances are affected by ManyEars' ability to separate sound sources in such extreme conditions, a factor that goes beyond our object of study, which is to evaluate the feasibility and advantages of integrating an artificial audition system to a dialogue management system.

The robot remained immobile during the trials, to avoid having to deal with the complexity that brings mobility for a posteriori analysis of dialogues. Recognition performances were derived for the system using ManyEars, and from audio streams generated by using the

signal coming from one microphone. This makes it possible to evaluate the added capabilities provided by ManyEars.

A total of 2343 audio streams were recorded during these trials using ManyEars (1488) and using one microphone (855). They were then categorized based on audio stream quality (by listening to them to validate subjectively the quality of the recorded speech) and by looking at what resulted from the Dialogue Management module (i.e., Successful or Unsuccessful recognition). Stream quality is characterized by five types :

- **Good** : the audio stream contains audible speech with no imperfections.

- **Noisy** : the audio stream contains audible speech but with a pitch boost (e.g., a sudden noise) or similar inconsistencies that affect recognition. Also, the text-to-speech process is independent of audio processing, and we tried to avoid listening when the robot is talking. However, without having feedback on when the robot is done talking, in a small number of cases, audio streams are corrupted with the robot's own voice.

- **Duplicated** : with ManyEars, sometimes the same sound is detected in two different locations (by the reflection of sound on an object), generating two distinct but quasi-identical audio streams. This may affect the performance of the Dialogue Management module (which is influenced by the lexicon, which in turn depends on the what has been recognized and leading to a state change on the robot).

- **Useless** : the audio stream contains audible speech that cannot be used to influence the robot's state, and therefore must not result in successful recognition by the Dialogue Management module.

Table 4.1 and Table 4.2 summarize the performances observed using ManyEars and using one microphone, respectively. For each combination of stream quality and trials conditions, the numbers n of associated audio streams are presented, along with the ratio of Dialogue Management results (Successful or Unsuccessful) associated with the audio stream quality type. *Rel* refers to the overall performance observed relative to stream quality, while *Abs* relates to the absolute performance observed for the audio streams processed using ManyEars or using one microphone. Note that for the one microphone case, the Duplicated type is not observed because this phenomenon occurs only with ManyEars. The objective is for the system to process successfully streams with audible speech (Good and Noisy), and otherwise not result in successful recognition (Useless).

Tableau 4.1 Dialogue Management performances using ManyEars

Stream Quality	Cnd	n	Success	Unsuccess
Good	LAB	351	77.8	22.2
	CAF	46	50	50
	Rel	397	74.6	25.4
	Abs	1488	19.8	6.8
Noisy	LAB	167	79.6	20.4
	CAF	42	54.8	45.2
	Rel	209	74.6	25.4
	Abs	1488	10.6	3.6
Duplicated	LAB	186	58.1	41.9
	CAF	10	0	100
	Rel	196	55.1	44.9
	Abs	1488	7.2	6
Incorrect	LAB	162	21	79
	CAF	39	12.8	87.2
	Rel	201	19.4	80.6
	Abs	1488	2.7	10.9
Useless	LAB	244	0	100
	CAF	241	0	100
	Rel	485	0	100
	Abs	1488	0	32.6
Overall	-	1488	40.3	59.7

Tableau 4.2 Dialogue Management performances using one microphone

Stream Quality	Cnd	n	Success	Unsuccess
Good	LAB	413	29.5	70.5
	CAF	8	0	100
	Rel	421	29	71
	Abs	855	14.2	35.1
Noisy	LAB	40	5	95
	CAF	3	0	100
	Rel	43	4.7	95.3
	Abs	855	0.3	4.7
Incorrect	LAB	137	11.7	88.3
	CAF	27	0	100
	Rel	164	9.8	90.2
	Abs	855	1.9	17.3
Useless	LAB	136	0	100
	CAF	91	0	100
	Rel	227	0	100
	Abs	855	0	26.6
Overall	-	855	16.4	83.6

For Good and Noisy audio streams processed by ManyEars, results are similar : the system successfully recognized 74.6%, which is much better than what is achieved with one microphone (29.5% for Good and 5% for Noisy audio streams, in lab conditions only). For Noisy audio streams, even though ManyEars introduced some unwanted distortions in 13.9% (10.6%+3.6%) of the audio streams, CSLU was able to process 79.5% of them in controlled conditions (and 54.8% in the cafeteria). Useless audio streams from ManyEars are also processed correctly by not leading to false positive. Therefore, considering successful recognition for Good and Noisy audio streams, and unsuccessful recognition for Useless audio streams, the system performs adequately for 63% over 73.4% of the audio streams generated by ManyEars, compared to 41.1% over 80.8% for audio streams derived using one microphone.

For the Duplicated audio streams, while a good proportion of successful recognition is observed in lab conditions, none is observed in the cafeteria. In such an open settings, vocal interactions occur faster and state change of the robot happens more often. We observe that Duplicated audio streams occur mainly in scenarios where a single interlocutor with a loud voice is interacting with Spartacus precisely when the environment becomes quieter. ManyEars then detects two different sound sources, which sometimes lead to unpredictable

behavior for the dialogue manager. This is something to improve on ManyEars by adding for instance the ability to identify the sound sources before sending it to recognition and process by the dialogue manager. For the trials conducted, this would have resulted in a 6% improvement. For the Incorrect audio streams, CLSU's repair feature led to the recovery of only a small proportion of the vocal interactions, for both ManyEars and the one microphone setup. It may not be as beneficial as expected, because it can lead to false-positives.

As expected, recognition performances are better in the lab environment compared to the cafeteria. This can be explained because ManyEars has difficulty tracking and separating many sound sources simultaneously in such noisy conditions, leading to incomplete audio streams and the introduction of inconsistencies. However, in spite of the very difficult conditions, the system was able to get successful recognition in the cafeteria (e.g., 50% of Good audio streams, 54.5% of Noisy audio streams, and even 12.8% of Incorrect audio streams), while none were recognized using one microphone. The problem comes from the inability to discriminate sound sources in noisy conditions using only one microphone. For instance, during a six minute trial in the cafeteria, only 12 audio streams (compared to 103 using ManyEars) were recorded, with an average length of 26 sec (the longest one lasting 108 sec). All users were recorded on top of each other, resulting in long incomprehensible audio streams for speech recognition, even hard to interpret for a human listener. Recording would start when someone would begin talking and would be performed until a total silence would occur around Spartacus.

With an overall result of 40.3% successful recognition using ManyEars compared to 16.4% with one microhone, the integration of ManyEars to CSLU brought significant improvement but still needs to be improved. However, considering that unsuccessful recognition of Useless audio streams is coherent with what the system has to do in these cases, we can consider the overall performance of the integration to be 72.9% $(40.3 + 32.6)$ compared to 43% $(16.4 + 26.6)$ with the use of one microphone, and with successfull recognition of 59.8% $(40.3/(100 - 32.6))$ compared to 22.3% $(16.4/(100 - 26.6))$ respectively.

4.6 Conclusion

This paper presents the integration of ManyEars, a sound source localization, tracking and separation pre-processing system, with CLSU dialogue management system, to see how it can affect natural language processing performances. Results from our trials suggest that such integration improves dialogue management performances in challenging conditions,

in comparison with the direct use of speech input coming from a regular microphone. The results are encouraging, but there are still improvements to be made to reach adequate performances for natural language processing. In future work, ManyEars will be improved to provide better quality in open and complex settings. For instance, we are currently working on an approach to dynamically adapt the input gain of ManyEars to minimize distortions presented in the separated audio streams. Speaker identification will also be added to facilitate tracking and identification of sound sources.

Acknowledgment

F. Michaud holds the Canada Research Chair (CRC) in Mobile Robotics and Autonomous Intelligent Systems. Support for this work is provided by the Natural Sciences and Engineering Research Council of Canada, the Canada Research Chair program and the Canadian Foundation for Innovation.

CHAPITRE 5
CONCLUSION

Ce mémoire présente l'intégration du gestionnaire de dialogues CSLU au système d'audition artificielle ManyEars et à la plateforme robotique Spartacus ayant à évoluer dans un environnement ouvert de manière autonome. Le système résultant est fonctionnel et fut validé lors de la conférence scientifique AAAI et lors d'essais plus spécifiques en laboratoire et dans un milieu public. Il en résulte un système de traitement du langage naturel flexible, permettant une adaptation dynamique et en temps réel de la grammaire à utiliser selon le contexte dans lequel le robot évolue et ce, par l'échange des éléments contenus dans les scénarios de dialogue et les interfaces graphiques de Spartacus. Cette approche procure une très grande versatilité et permet de faire évoluer dynamiquement les scénarios de dialogues du robot. De plus, il est désormais possible de donner des instructions à Spartacus en language naturel pour l'aider à évoluer dans sa prise de décisions et également de suivre facilement son évolution dans le temps. Les données audio enregistrées et filtrées par ManyEars sont transférées au gestionnaire de dialogue afin d'effectuer la reconnaissance vocale, d'analyser les interprétations et de prendre des décisions selon l'état du robot. Aussi, afin d'assurer un suivi constant de la progression du robot dans le temps, le système émet des trames sonores en synthétisant une voix et affiche l'évolution de Spartacus sur les interfaces graphiques de l'écran tactile posé sur le torse du robot. Ceci est possible grâce à l'échange de données entre le module décisionnel et comportemental du robot avec le gestionnaire de dialogues. Enfin, en plus de démontrer la faisabilité d'intégrer efficacement un gestionnaire de dialogues à un système d'audition artificielle, le projet démontre que ManyEars améliore sensiblement les performances de reconnaissance de la parole.

Malgré tout le potentiel que démontre le système developpé, il reste encore des améliorations à apporter au système, principalement au niveau du système d'audition artificielle. En effet, dépendamment des situations, il arrive que le système d'audition artificielle duplique les sources sonores détectées ou qu'il ait de la difficulté à séparer l'ensemble des sources sonores obtenues dans des environnements bruités. L'audition artificielle en milieu naturel est un grand défi, et les travaux se poursuivent afin d'améliorer le tout. Un couplage plus fort entre le gestionnaire de dialogue et le système d'audition artificielle aiderait certainement à améliorer les performances globales, une piste à explorer davantage dans les travaux futurs.

ANNEXE A

CONCEPTION ITÉRATIVE DE ROBOTS MOBILES ÉVOLUÉS

Auteurs et affiliations :

- F. Michaud : Professeur, Laboratoire de robotique intelligente, interactive, intégrée et interdisciplinaire (IntRoLab), Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.
- D. Létourneau : Professionel de recherche, Laboratoire de robotique intelligente, interactive, intégrée et interdisciplinaire (IntRoLab), Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.
- E. Beaudry : Étudiant au doctorat, Université de Sherbrooke, Faculté des sciences, Département d'informatique.
- M. Fréchette : Étudiant à la maîtrise, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.
- F. Kabanza : Professeur, Laboratoire de planification en intelligence artificielle (Planiart), Université de Sherbrooke, Faculté des sciences, Département d'informatique.
- M. Lauria : Professeur, Laboratoire de robotique intelligente, interactive, intégrée et interdisciplinaire (IntRoLab), Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Date d'acceptation : Avril 2008

État de l'acceptation : version finale publiée

Journal : International Journal of Computing and Information Technology, Special Issue on Advanced Mobile Robotics

Référence : [Michaud *et al.*, 2009]

Titre français : Conception itérative de robots mobiles évolués

Résumé français : L'intégration de composantes matérielles, logicielles et décisionnelles est fondamentale dans l'élaboration d'un robot mobile avancé capable d'accomplir des tâches complexes dans un environnement non structuré et imprévisible. Nous réalisons ce défi en adoptant une stagégie d'innovation interactive centrée sur une architecture décisionnelle fondée sur la notion de sélection de modules comportementaux. Cette architecture a évolué au fil du temps, passant de l'intégration d'un système d'évitement d'obstacles à la lecture de messages, de l'ajout d'interfaces graphiques tactiles à l'intégration de la capacité de localisation et la cartographie à la planification de tâches, de la séparation et le suivi de source sonores à l'ajout d'un système de reconnaissance vocale, de gestion du dialogue et

de synthèse vocale. Conçu pour être un journaliste scientifique, le robot interagit de manière compréhensible et reconfigurable et fournit des intentions et des informations claires sur son évolution dans le contexte d'une conférence, soit instantanément ou lors d'analyses a posteriori. Cet article présente l'intégration de ces capacités sur le robot, relevant ainsi quelques nouvelles problématiques concernant la planification, la coordination des capacités des modules reliés à l'audition, au visuel ainsi qu'aux graphiques, et finalement l'utilité et l'impact du système décisionnel du robot dans un environnement d'opération non contrôlé. Cet article fait aussi mention de nouveaux besoins pour la prochaine itération de ce robot interactif, ajoutant de la compliance au système de locomotion et de manipulation ainsi qu'aux interactions naturelles en faisant usage des bras, des expressions faciales ainsi que la posture du robot.

A.1 Abstract

Integration of hardware, software and decisional components is fundamental in the design of advanced mobile robotic systems capable of performing challenging tasks in unstructured and unpredictable environments. We address such integration challenges following an iterative design strategy, centered on a decisional architecture based on the notion of motivated selection of behavior-producing modules. This architecture evolved over the years from the integration of obstacle avoidance, message reading and touch screen graphical interfaces, to localization and mapping, planning and scheduling, sound source localization, tracking and separation, speech recognition and generation on a custom-made interactive robot. Designed to be a scientific robot reporter, the robot provides understandable and configurable interaction, intention and information in a conference setting, reporting its experiences for on-line and off-line analysis. This paper presents the integration of these capabilities on this robot, revealing interesting new issues in terms of planning and scheduling, coordination of audio/visual/graphical capabilities, and monitoring the uses and impacts of the robot's decisional capabilities in unconstrained operating conditions. This paper also outlines new design requirements for our next design iteration, adding compliance to the locomotion and manipulation capabilities of the platform, and natural interaction through gestures using arms, facial expressions and the robot's pose.

A.2 Keywords

Integration, Decisional architecture, Task planning and scheduling, Human-robot interaction, Iterative design.

A.3 Introduction

The dream of having autonomous machines performing useful tasks in everyday environments, as unstructured and unpredictable as they can be, has motivated for many decades now research in robotics and artificial intelligence. However, intelligent and autonomous mobile robots is still in what can be identified as being research phases, i.e., mostly basic research and concept formulation, and some preliminary uses of the technology with attempts in clarifying underlying ideas and generalizing the approach [Shaw, 2002]. The challenge in making such a dream become a reality lies in the intrinsic complexities and interdependencies of the necessary components to be integrated in a robot. A mobile robot is a system with structural, locomotive, sensing, actuating, energizing and processing capabilities, which set the resources available to deal with the operating conditions of the environment. Such resources are exploited and coordinated based on decisional processes and higher cognitive functions, according to the intended tasks.

In his Presidential Address at the 2005 AAAI (American Association for Artificial Intelligence) Conference, Ronald Brachman argued that Artificial Intelligence (AI) is a system science that must target working in the wild, messy world [Brachman, 2006]. This was the initial goal of AI, which revealed to be too difficult to address with the technology

50 years ago, leading to the creation of more specialized subfields. The same can be said about the Robotics scientific community. The underlying complexities in these disciplines are difficult enough to resolve that a "divide-and-conquer" approach is justified. This explains why only a few initiatives around the world tackle directly this challenge. But those that do always identify new ways of solving the integration issues, outlining guidelines and specifications for new research questions or for new technologies aiming to improve robotic capabilities. Continuous technological progress makes it possible to push the state-of-the-art in robotics closer to its use in day-to-day applications, and it is by addressing directly the integration challenges that such objective can and will be accomplished.

Designing a mobile robot that must operate in public settings probably addresses the most complete set of issues related to autonomous and interactive mobile robots, with system integration playing a fundamental role at all levels. A great variety of robots have been and are currently being designed to operate in natural settings, interacting with humans in the wild, messy world. For the most recent and pertinent related work to our project, we can identify **non-mobile child-size humanoid** robots (e.g., Kaspar, from the RobotCub project, to investigate the use of gestures, expressions, synchronization and imitation ; Huggable [Stiehl et Breazeal, 2005], equipped with full-body sensate skin, silent voice coil-type electromagnetic actuators with integrated position, velocity and force sensing means) ; **adult-size humanoid torso** (e.g., Dexter [Brock *et al.*, 2005], for grasping and manipulation using tactile load cells on the finger, investigating decomposition-based planning and knowledge acquisition and programming by demonstration ; Domo [Edsinger-Gonzales et Weber, 2004], using stereo vision, force sensing compliant actuators (FSCA) and series elastic actuators (SEA) [Pratt et Williamson, 1995] to investigate alternative approaches to robot manipulation in unstructured conditions) ; **differential-drive platforms with manipulators** (e.g., B21, PeopleBot and Care-O-Bot platforms equipped usually with one robotic manipulator) ; **differential-drive mobile torsi** (e.g., Robonaut, from NASA, installed on a Segway Robotic Mobility Platform base and teleoperated ; Battlefield Extraction-Assist Robot - BEAR - from Vecna Robotic-US, using leg-track/dynamic balancing platform to carry fully-weighted human casualities from a teleoperation base) ; **omnidirectional humanoid base** (e.g., ARMAR III [Asfour *et al.*, 2006], using load cells, torque sensors, tactile skin, stereo vision and six microphone array in the head and a three-tier (planning, coordination, execution) control architecture to accomplish service tasks in a kitchen ; HERMES [Bischoff et Graefe, 2004], combining stereo vision, kinesthetic, tactile array bumper and single omnidirectional auditory sensing with natural spoken language (input via voice, keyboard or e-mail) with a situation-oriented skill-based architecture).

Even though each of these robots present interesting functionalities in terms of motion (omnidirectional, articulated torso) and interaction skills (gesture, expressions, tactile, force-controlled actuators, stereo vision, natural language, manipulate objects and collaborate with humans in task-oriented applications), they do not go far in terms of cognition (e.g., autonomous decisions in open settings, adaptation to a variety of situations). They also make compromises on the advanced capabilities of the integrated elements : not all of them use state-of-the-art technologies. These platforms are however necessary in making significant contributions for moving toward higher level of cognition (e.g., future research

with Domo targets the implementation of a motivational system with homeostatic balance between basic drives and a cognitive layer, plus incremental learning of sensorimotor experiences [Edsinger-Gonzales et Weber, 2004]).

In opposition, our work initiated from cognitive architectural considerations, and moved to software and hardware requirements of a mobile robot operating in real life settings. EMIB (Emotion and Motivation for Intentional selection and configuration of Behavior-producing modules) was our first computational architecture used to integrate all necessary decisional capabilities of an autonomous robots [Michaud, 2002]. It was implemented on a Pioneer 2 robot entry in the 2000 AAAI Mobile Robot Challenge. A robot had to start at the entrance of the conference site, find the registration desk, register, perform volunteer duties (e.g., guard an area) and give a presentation [Maxwell *et al.*, 2004]. Our team was the first and only one to attempt the event from start to end, using sonars as proximity sensors, navigating in the environment by reading written letters and symbols using a pan-tilt-zoom (PTZ) camera, interacting with people through a touch-screen interface, displaying a graphical face to express the robot's emotional state, determining what to do next using a finite-state machine (FSM), recharging itself when needed, and generating a HTML report of its experience [Michaud *et al.*, 2001]. We then started to work on a more advanced platform that we first presented at the 2005 AAAI Mobile Robot Competition [Michaud *et al.*, 2005a, 2007]. Our focus was on integrating perceptual and decisional components addressing all four issues outlined during the AAAI events over the years, i.e. : 1) the robust integration of different software packages and intelligent decision-making capabilities ; 2) natural interaction modalities in open settings ; 3) adaptation to environmental changes for localization ; and 4) monitoring/reporting decisions made by the robot [Gockley *et al.*, 2004; Maxwell *et al.*, 2004; Simmons *et al.*, 2003; Smart *et al.*, 2003]. Trials conducted at the 2005 AAAI event helped established new requirements for providing meaningful and appropriate modalities for reporting and monitoring the robot's states and experiences.

When operating in open settings, interactions are rapid, diverse and context-related. Having an autonomous robot determine on its own when and what it has to do based on a variety of modalities (time constraints, events occurring in the world, requests from users, etc.) also makes it difficult to understand the robot's behavior just by looking at it. Therefore, we concentrated our integration effort in 2006 on designing a robot that can interact and explain, through speech and graphical displays, its decisions and its experiences as they occur (for on-line and off-line diagnostics) in open settings. This paper first presents the software/hardware components of our 2006 AAAI robot entry, its software and decisional architectures, the technical demonstration made at the conference along with the interfaces developed for reporting the robot's experiences. This outlines the progress made so far in addressing the integration challenge of designing autonomous robots. The paper then describes the design specifications of our new advanced mobile robot platform, derived from what we have identified as critical elements in making autonomous systems operating in everyday environments.

A.4 A scientific robot reporter

Our 2006 AAAI robot entry, shown in Figure A.1, is a custom-built robot equipped with a SICK LMS200 laser range finder, a Sony SNC-RZ30N 25X pan-tilt-zoom color camera, a Crossbow IMU400CC inertial measurement unit, an array of eight microphones placed in the robot's body, a touch screen interface, an audio system, one on-board computer and two laptop computers. The robot is equipped with a business card dispenser, which is part of the robot's interaction modalities. A small camera (not shown in the picture) was added on the robot's base to allow the robot to detect the presence of electric outlets. The technique used is based on a cascade of boosted classifiers working with Haar-like features [Lienhart et Maydt, 2002] and a rule-based analysis of the images.

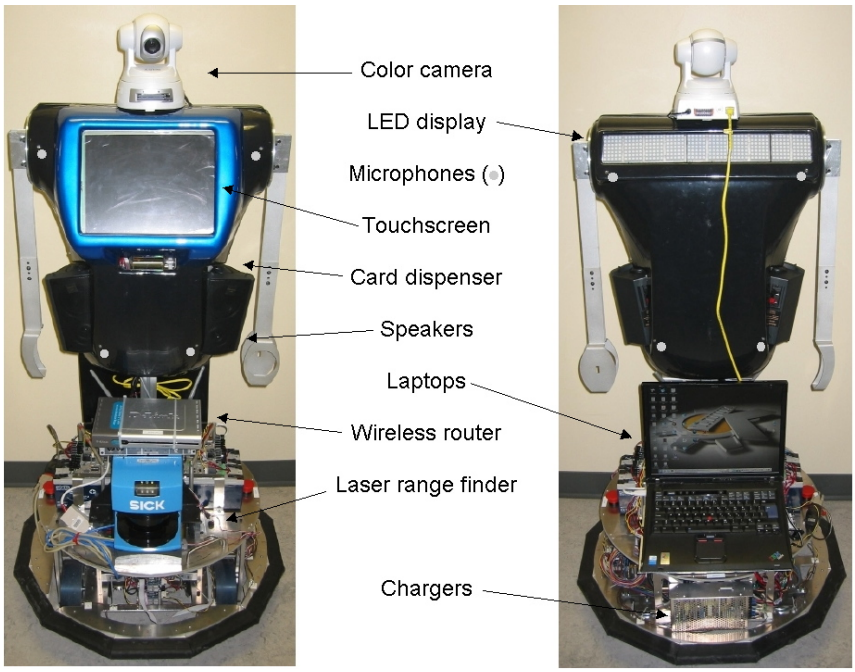

Figure A.1 Our 2006 AAAI robot entry (front view, back view).

Figure A.2 illustrates the robot's software architecture. It integrates Player for sensor and actuator abstraction layer [Vaughan *et al.*, 2003] and CARMEN (Carnegie Mellon Robot Navigation Toolkit) for path planning and localization [Montemerlo *et al.*, 2003]. Also integrated but not shown on the figure is Stage/Gazebo for 2D and 3D simulators, and Pmap library[1] for 2D mapping, all designed at the University of Southern California. RobotFlow and FlowDesigner (FD) [Cote *et al.*, 2004] are also used to implement the behavior-producing modules, the vision modules for reading messages [Letourneau *et al.*, 2004] and SSLTS, our real-time sound source localization, tracking [Valin *et al.*, 2007a] and separation [Valin *et al.*, 2004] system. For speech recognition and dialogue management, we interfaced the CSLU toolkit (http ://cslu.cse.ogi.edu/toolkit/). One key feature of CSLU is that each grammar can be compiled at runtime, and it is possible to use dynamic grammars that can change on-the-fly before or while the dialogue system is running.

[1] http ://robotics.usc.edu/ ahoward/pmap/

Software integration of all these components are made possible using MARIE, a middleware framework oriented towards developing and integrating new and existing software for robotic systems [Cote *et al.*, 2006a,b].

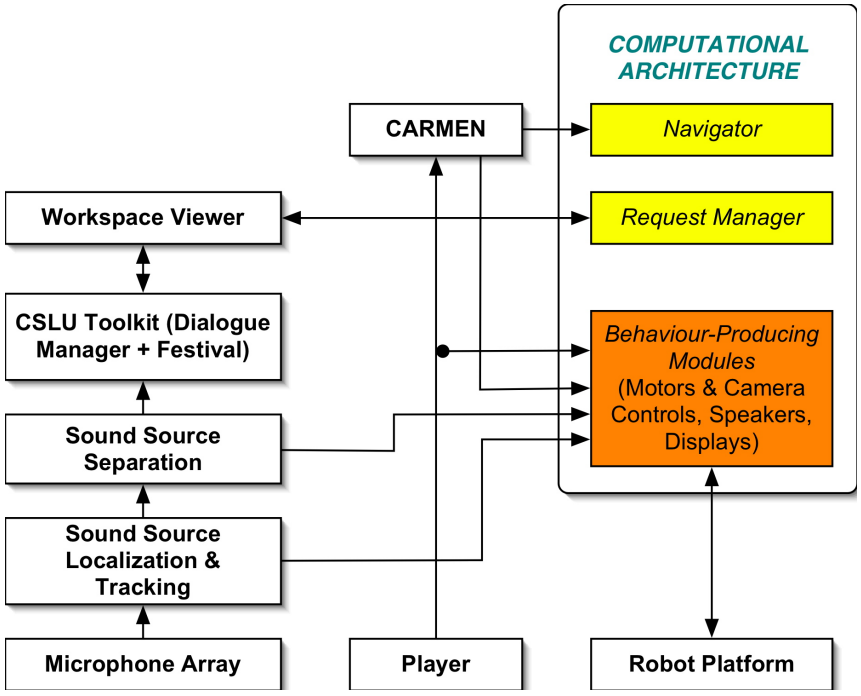

Figure A.2 Our robot's software architecture.

A.4.1 Decisional architecture and modules

The robot is programmed to respond to requests from people, and with only one intrinsic goal of wanting to recharge when its energy is getting low (by either going to an outlet identified on the map or by searching for one, and then asking to be plugged in). Our robot's duty is to address these requests to the best of its capabilities and what is 'robotically' possible. Such requests may be to deliver a written or a vocal message to a specific location or to a specific person, to meet at a specific time and place, to schmooze, etc. The robot may receive multiple requests at different periods, and has to autonomously manage what, when and how it can satisfy them.

With our robot, we assume that the most appropriate approach for making a robot navigate in the world and accomplish various tasks is done through the use of behavior-producing modules (also known as behavior-based systems [Arkin, 1998]). As a result, representation and abstract reasoning working on top of these modules become a requirement, with the objective of being able to anticipate the effects of decisions while still adapt to the inherently complex properties of open and unconstrained conditions of natural living environments.

The decisional architecture developed for our robot's decision-making capabilities is shown in Figure A.3. It is based on the notion of having motivated selection of behavior-producing

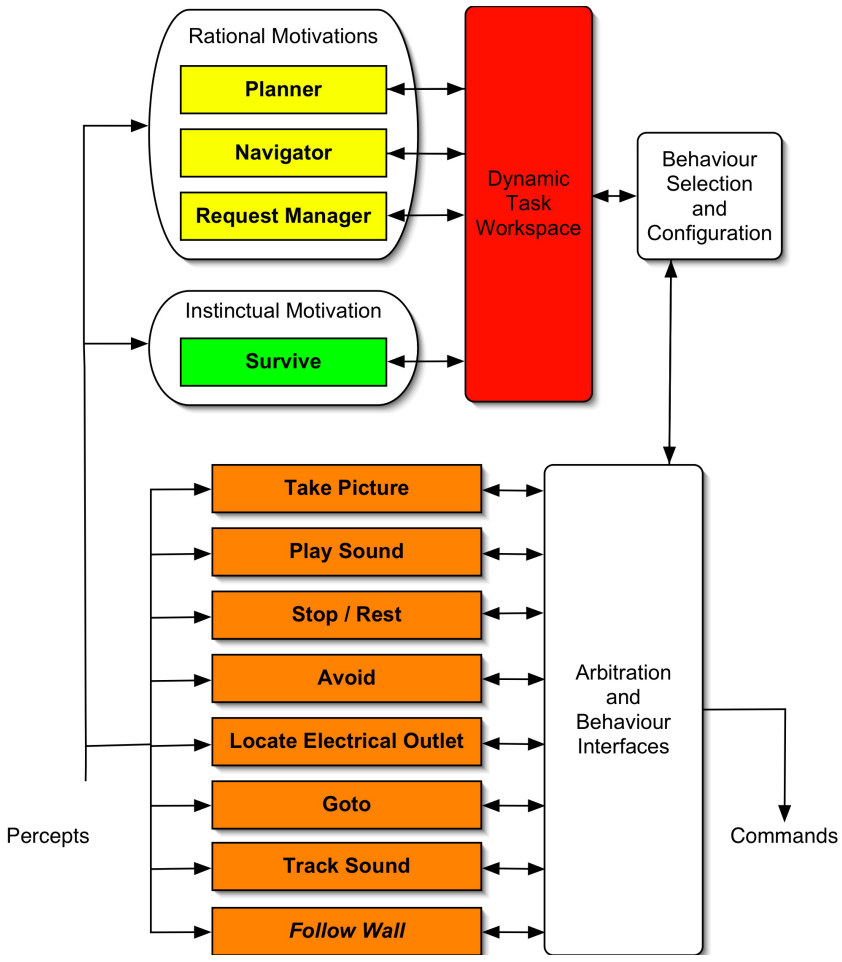

Figure A.3 Our robot's decisional architecture.

modules. We refer to it as MBA, for Motivated Behavioral Architecture [Michaud *et al.*, 2005a, 2007]. It is composed of three principal components :

1. Behavior-producing modules (BPMs) define how particular percepts and conditions influence the control of the robot's actuators. Their name represents their purpose. The actual use of a BPM is determined by an arbitration scheme realized through the Arbitration and Behaviour Interfaces module (implementing in our case a priority-based / subsumption arbitration), and the BPM activation conditions derived by the Behaviour Selection and Configuration module (explained later).

2. Motivational sources (or Motivations, serving to make the robot do certain tasks) recommend the use or the inhibition of tasks to be accomplished by the robot. Motivational sources are categorized as either instinctual, rational or emotional. This is similar to considering that the human mind is similar to a "committee of the minds," with instinctual, rational, emotional, etc. minds competing and interacting [Werner, 1999]. Instinctual motivations provide basic operation of the robot using simple rules. Rational motivations are more related to cognitive processes, such as navigation and planning. Emotional motivations monitor conflictual or transitional situations between tasks. The importance of each of these influences explains why manifested behavior can be more influenced by direct perception, by reasoning or by managing commitments and choices. By distributing motivational sources and distinguishing their roles, it is possible to exploit more efficiently the strengths of various influences regarding the tasks the robot must accomplish, while not having to rely on only one of them for the robot to work. This is similar in essence to the basic principles of behavior-based control, applied at a higher abstraction level.

3. Dynamic Task Workspace (DTW) serves as the interface between motivational sources and BPMs. Through the DTW, motivations exchange information asynchronously on how to activate, configure and monitor BPMs. The interface between the BPMs and the motivational sources is done through tasks in the DTW. Tasks are data structures that are associated with particular configuration and activation of one or multiple behaviors. The DTW organizes tasks in a tree-like structure according to their interdependencies, from high-level/abstract tasks (e.g., 'Deliver message'), to primitive/BPM-related tasks (e.g., 'Avoid'). Motivations can add and modify tasks by submitting modification requests or queries, or subscribe to events regarding the task's status.

By exchanging information through the DTW, motivations are kept generic and independent from each other, allowing motivations to distributively come up with behavior configuration based on the capabilities available to the robot. For instance, one instinctual motivational source may monitor the robot's energy level to issue a recharging task in the Dynamic Task Workspace, which activates a 'Locate Electrical Outlet' behavior that would make the robot detect and dock in a charging station. Meanwhile, if the robot knows where it is and can determine a path to a nearby charging station, a path planning rational motivation can add a subtask of navigating to this position, using a 'Goto' behavior. Otherwise, the 'Locate Electrical Outlet' behavior will at least allow the robot to recharge opportunistically, when the robot perceives a charging station nearby.

Only instinctual and rational motivations are considered in this study, with rational motivations having greater priority over instinctual ones in case of conflicts. *Survive* makes the robot move safely in the world while monitoring its energy level. *Planner* determines which primitive tasks and which sequence of these tasks are necessary to accomplish higher-level tasks under temporal constraints and the robot's capabilities (as defined by BPMs). This motivational source is detailed in the next section. *Navigator* determines the path to a specific location to accomplish tasks posted in the DTW. *Request Manager* handles task requested by the users via the touch screen interface or from vocal commands. Task requests are then processed by *Planner* and added to the actual plan if appropriate.

With multiple tasks being issued by the motivational sources, the Behavior Selection and Configuration module determines which behaviors are to be activated according to recommendations made by motivational sources, with or without particular configuration (e.g., a destination to go to). A recommendation can either be negative, neutral or positive, or takes on real values within this range regarding the desirability of the robot to accomplish specific tasks. Activation values reflect the resulting robot's intentions derived from interactions between the motivational sources. Behavior use and other information coming from behavior and that can be useful for task representation and monitoring are also communicated through the Behavior Selection and Configuration module.

Task planning motivation

This motivational source invokes, when a new task is placed in the DTW, a planning algorithm that combines principles from SHOP2 HTN planning algorithm [Nau *et al.*, 2003] and SAPA [Do et Kambhampati, 2003]. As in SHOP2, we specify a planning domain (e.g., the domain of attending a conference at AAAI) by describing the robot primitive behaviors in terms of template tasks and methods for recursively decomposing template tasks down to primitive tasks which are atomic actions. For instance, we can specify that the task of making a presentation at location px is from time t_1 to time t_2, and that it can be decomposed into the simpler subtasks of going to px and presenting at time t_1. Decomposition of tasks into simpler ones are given with preconditions under which the decomposition is logically sound and time constraints for ensuring its consistency. Given such a set of task specifications, the planning algorithm consists of searching through the space of tasks and world states.

More specifically, starting from a given set of initial tasks and a current state describing the robot state and environment situation, the planner explores a space of nodes where each node represents : the current list of subtasks ; the current robot state and environment situation ; the current plan (initially empty). A current node is expanded into successors by decomposing one of the current task into simpler ones (using the specified task decomposition method) or by validating a current primitive task against the current world state (this is done by checking its precondition and updating the current world state using the primitive task's effects) and adding it to the current plan. The search process stops when a node is reached with no more task to decompose. On a node, there can be different ways of decomposing a task and different orders in which to decompose them ; backtracking is invoked to consider the different alternatives until obtaining a solution. This can be a source of exponential growth during the search process. By carefully engineering the

decomposition methods to convey some search control strategy, it is possible to limit this state explosion [Nau *et al.*, 2003].

Task Planning with Temporal Constraints. A normal HTN planning algorithm does not consider temporal constraints [Nau *et al.*, 2003, 2004]. To implement this, we modified the basic HTN planning algorithm by : (a) adding a *current-time* variable into the representation of a current state during search ; (b) specifying time constraints in the specification based on this variable in the precondition of task decomposition methods and of primitive tasks ; (c) allowing the specification of conditional effect update for primitive tasks based on this variable. That way, we extend the HTN concept to support time constrained tasks by adding temporal constraints at each decomposition level. The planner can use these temporal constraints to add partial orders on tasks. These partial orders reduce the search space and accelerate the plan generation process. These temporal constraints can also be transferred when a high-level task is decomposed into lower-level tasks.

When defining temporal intervals in the domain specification, care must be taken to establish a good compromise between safety and efficiency for task execution. Being too optimistic may cause plan failure because of lack of time. For instance, assuming that the robot can navigate at high speed from one location to the other will cause a plan to fail if unpredictable events slow down the robot. To solve this problem, the solution is to be conservative in the domain model and assume the worst case scenario. On the other hand, being too conservative may lead to no solution. Therefore, temporal intervals in the domain model are specified using an average speed (0.14 m/s) lower than the real average speed (0.19 m/s) of the robot.

Over the last years, efforts has been done in creating formal techniques for planning under temporal and resource uncertainty [Bresina *et al.*, 2002]. To keep our approach simple and to avoid having to deal with complex models of action duration, we use the following heuristics : planning is initiated first using a conservative action duration model and, when a possible opportunity (using the temporal information associated with the tasks) is detected, the planner is reinvoked to find a better plan. In other words, if the robot proceeds faster than what is planned (i.e., the end of the updated projected action is lower than the end of the planned action), it is possible that a better plan exists, and replanning therefore occurs.

Time Windowing. Using a technique borrowed from SAPA [Do et Kambhampati, 2003], our planner post-process the plan generated by the search process to obtain a plan with time windowing for actions. During search, each node is associated with a fixed time stamp (that is, the $current - time$ variable), and at the end of the search process we obtain a plan consisting of a sequence of actions each assigned with a time stamp indicating when it should be executed. This sequence of actions is post-processed based on time constraints in the domain specification (i.e., constraints attached to task decomposition methods and primitive tasks) to derive time intervals within which actions can be executed without jeopardizing the correctness of the plan.

Task Filtering and Priority Handling In a traditional planning setting, a list of initial tasks is considered as a conjunctive goals. If the planner fails to find a plan that achieves them all, it reports failure. In the context of the AAAI Challenge as well as in many real life situations, we expect the robot to accomplish as many tasks as possible, with some given preferences among task. For instance, the robot may fail to deliver one message at the right place, but successfully deliver another one. This would be acceptable depending on the environment conditions.

We implement a robot mission as a list of tasks, each associated with a degree of preference or priority level. Then we iteratively use the HTN planner to obtain a plan for a series of approximation of the mission, with decreasing level of accuracy. Specifically, initially we call the planner with the entire mission ; if it fails to compute a plan within a deadline set empirically in the robot architecture (typically 1 second), a lowest priority task is removed from the mission and the planner is called with the remaining mission. This is repeated until a solution plan is found. If the mission becomes empty before a solution is found, failure is returned (the entire mission is impossible). Clearly, this model can be easily configured to include vital tasks that cause failure whenever any of them is not achievable. We also use some straightforward static analysis of the mission to exclude for instance low priority tasks that apparently conflict with higher priority ones.

Anytime Task Planning. Similarly to SAPA [Do et Kambhampati, 2003] and SHOP2 [Nau *et al.*, 2003], our planner has an anytime planning capability, which is important in mobile robotic applications. When a plan is found, the planner tries to optimize it by testing alternative plans for the same mission until the empirical maximum planning time is reached.

On-line Task Planning. The integration of planning capabilities into a robotic system must take into account real-time constraints and unexpected external events that conflict with previously generated plans. To reduce processing delays in the system, our planner is implemented as a library. The MARIE application adapter that links the planner to the rest of the computational architecture loads the planner library, its domain and world representation (e.g., operators, preconditions and effects) at initialization. The planner remains in memory and does not have to be loaded each time it is invoked. A navigation table (providing the distances from each pairs of waypoints) also remains in memory and is dynamically updated during the mission. External events are accounted for by monitoring the execution of a plan and by validating preconditions and time constraints of remaining actions in the plan. Violated preconditions activates re-planning.

The robot can receive task requests at any time during a mission. This requires the robot to update its plan even if it is not at a specified waypoint in its world representation. When generating a task plan, the duration of going from one waypoint to another is instantiated dynamically based on the current environment. That is, the duration of a *Goto(X)* action is not taken from the navigation table but is rather estimated from the actual distance to the targeted waypoint X, as provided by an internal path-planning algorithm. Therefore, *Planner* only has to deal with high-level navigation tasks, and it is not necessary to plan intermediate waypoints between two waypoints that are not directly connected. To estimate the duration for navigating from a waypoint to another, the planner

uses a navigation table of the size n^2 where n is the number of defined waypoints. This table is initialized by computing the minimum distance between waypoints with a classic A* algorithm. Each time a new waypoint is added on the map, the distances table is updated dynamically.

Interaction modalities

With a distributed cognitive architecture integrating a high number of capabilities to be used in open conditions, it is difficult to assess what is going on simply by looking at the robot's behavior, or by analyzing log files off-line. More dynamic and interactive tools are required. Here is the set of modalities designed to improve our understanding of integrated modalities (vision, audition, graphical, navigation) on our robot.

- **Visualization tools of decisional components.** The underlying objectives of such tools is to facilitate integration work by offering visualization tools of the decisional components added to the robot. It requires coherent and user-friendly representation of the integrated components so that developers can concentrate on their combined usage instead of spending time extracting and interpreting data in log files. In 2005, we developed a log viewing application, allowing us to monitor off-line the complete set of states of the robot through log files created by the different software components [Michaud *et al.*, 2007]. In 2006, we extended this tool to provide a dynamic and real-time view of what the robot is actually doing and planning to do, on-line. We call this updated version the WorkspaceViewer. With the WorkspaceViewer, it is now possible to display, directly on our robot's graphical interface, contextual information according to its current plan (e.g., behavioral configuration, map with dynamic places, active tasks). In addition, we can still use the basic off-line log viewing application to replay logs off-line.

  Figure A.4 shows a representation of the WorkspaceViewer's main window. The upper section contains a timeline view of DTW events (first line), planner events (second line), and behaviors' activations and exploitations (third line). The bottom section shows detailed information according to the position of the vertical marker on the timeline : a list of DTW's tasks and properties (first window from the left), active motivations (second window), the current plan (third window), the map of the environment and the trajectory of the robot (fourth window), the behaviors and their activations and exploitations (under the first window). The WorkspaceViewer is directly connected to the DTW and displays information as they become available in real-time. The WorkspaceViewer is also linked to *Request Manager* motivation to handle user requests.

- **Visualization of the SSLTS results.** Audio data is difficult to monitor in dynamic and open conditions : it is not persistent as a visual feature can be, for instance. An interface representing where the sound sources around the robot are detected and what was recorded by the robot revealed to be necessary from what we experienced during our 2005 participation. Figure A.5 illustrates the interface developed. The upper part shows the angle of the perceived sound sources around the robot, in relation to time. The interface shows in real-time the sound sources perceived, using

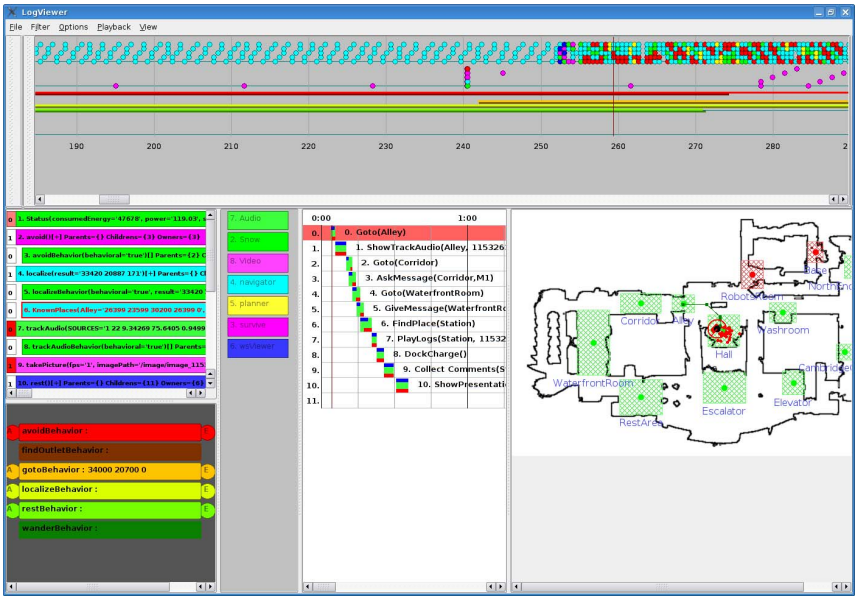

Figure A.4 WorkspaceViewer's main window.

dots of a distinct color. The sound sources are saved, and the user can select them and play back what the SSLTS generated. This interface reveals to be very valuable for explaining what the robot can hear, to construct a database of typical audio streams in a particular setting, and to analyze the performance of the entire audio module (allowing to diagnose potential difficulties that the speech recognition module has to face).

For instance, more than 6600 audio streams were recorded over the twelve hours of operation of the robot in open settings at the AAAI 2006 conference, held at Seaport Hotel and World Trade Center in Boston. From this set of audio streams, we observed that around 61% were audio content (people talking nearby the robot, sounds of different sort) adequately processed by our SSLTS system (i.e., the audio streams were correctly separated and located, and are understandable by a human listener). However, they were not related to specific interaction with the robot, and therefore revealed to be useless for interacting with our robot. 3.5% of the audio streams were inaudible by a human listener, and an additional 34% were either very short (e.g., caused by a noise burst or laughter) or truncated (e.g., when the robot starts talking, to avoid making the robot hear itself talk ; sound amplitude too low). Implementation problems caused these limitations, as we diagnosed after the event. This left around 1.5% of audio streams intended for specific interaction with the robot adequately processed by the SSLTS system. All were correctly processed using CLSU in specific interaction our robot had with human interlocutors. This clearly demonstrates the challenge of making an artificial audition system for operation in open conditions. Such auditory capability working in real-time in such conditions has not yet been demonstrated on other robots. The performance observed is impressive, but the challenges to overcome to make it more robust and efficient are still quite important. We will continue to improve SSLTS' performance by not conducting

speech recognition on audio streams that are too small (i.e., < 1 sec) and by making it possible to concurrently run multiple speech recognition processes (currently only one stream can be processed at a time).

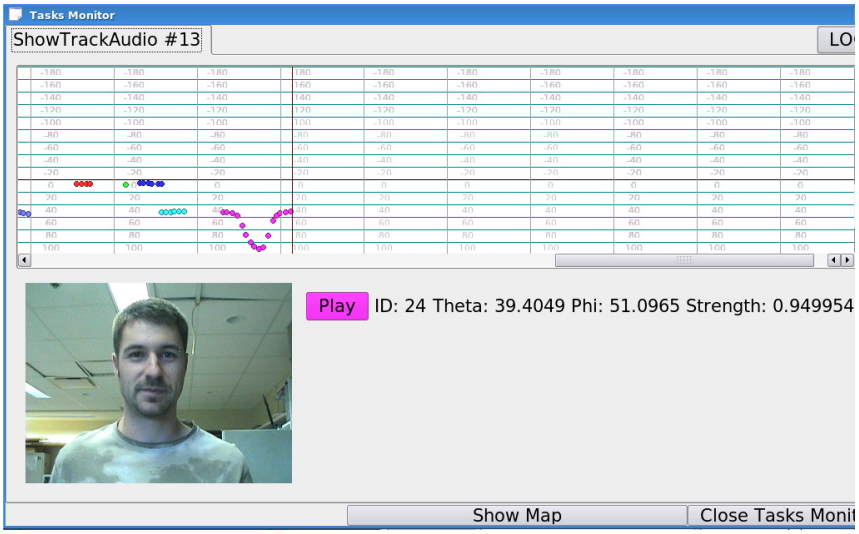

Figure A.5 Track audio interface.

- **Multimodal interaction interfaces.** It is sometimes difficult to know exactly which task is prioritized by the robot at a given time, making the interlocutors unable to know what the robot is actually doing and evaluate its performance. So we made the robot indicate its intention verbally and also graphically (in case, for whatever reasons, the interlocutor is not able to understand the message). Figure A.6 illustrates the graphical interface with which our robot is requesting assistance to find a location in the convention center.

  The graphical interfaces of the robot were made so that users can indicate through push buttons (using the touch screen) what they want our robot to do. We also made it possible for the users to make these requests verbally, saying out loud the names of the buttons. To facilitate the recognition process, specific grammar and dialogue managers are loaded in CSLU for each graphical mode. The audio and graphical interaction are therefore tightly integrated together.

A.4.2 Demonstration and results

Our robot demonstrated its capabilities in the AAAI 2006 Human-Robot Interaction Event, evaluated in a technical and a public demonstrations. Our robot entered five of the seven categories by doing the following :

- Natural Language Understanding and Action Execution : Following requests from humans, written or verbal, for making the robot do different tasks.
- Perceptual Learning Through Human Teaching : Learning of a location, specified by humans or identified by the robot (i.e., electrical outlet), and being able to remember it.

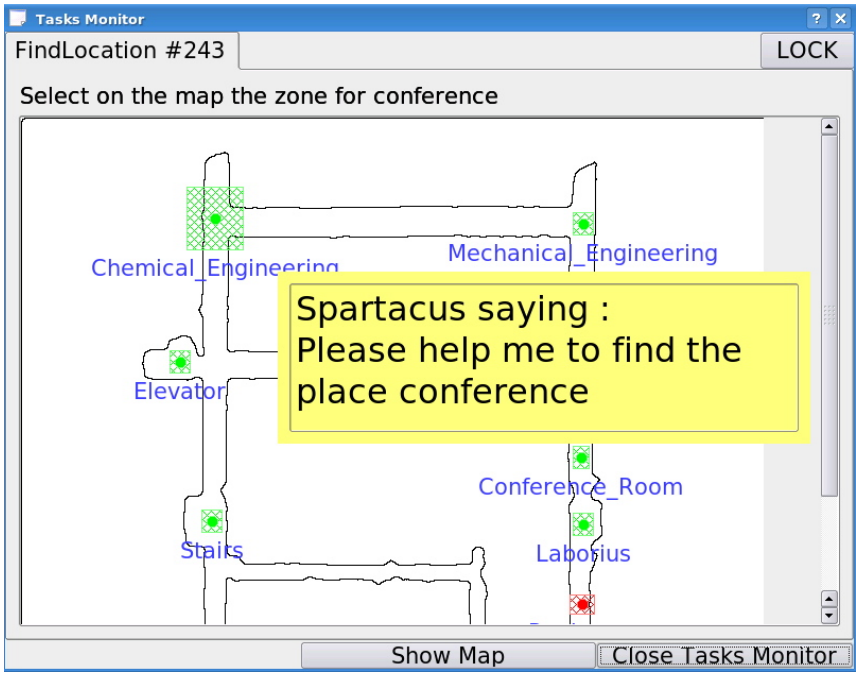

Figure A.6 Graphical display for an assistance request to find a location (in laboratory conditions).

- Perception, Reasoning, and Action : Temporal reasoning, planning and scheduling of tasks and intents.

- Shared Attention, Common Workspace, Intent Detection : Directing camera view in the direction of the speaker, communicate temporal constraints regarding task, and use of the dynamic (context-configured) viewer, on-line and off-line.

- Integration Challenge Categories 3 to 6 : Robot Scientific Reporter/Assistant demo, with understandable interaction, decision and situations experienced in unconstrained conditions.

Our technical demonstration, programmed to last 25 minutes (setting an overall fixed time constraint for the mission), consisted of five phases done in the area represented in Figure A.7 :

1. INTRODUCTION. Presentation (max 4 :30 min) at *Alley* of our robot's features : track audio viewer for separated sources ; directing camera view toward the speaker, with snapshots memorized every second ; graphic displays for user input ; context-based grammar ; pre-mapping of the area ; Gantt chart representation of the plan.

2. NAVIGATION. Go to *Corridor* to receive a message (voice attachment or text) to be delivered at *WaterfontRoom*. No time constraints are specified for this task. A specific graphical interface is displayed by the WorkspaceViewer to get the message. The *Planner* inserts this task if time permits between existing tasks (which are time constained).

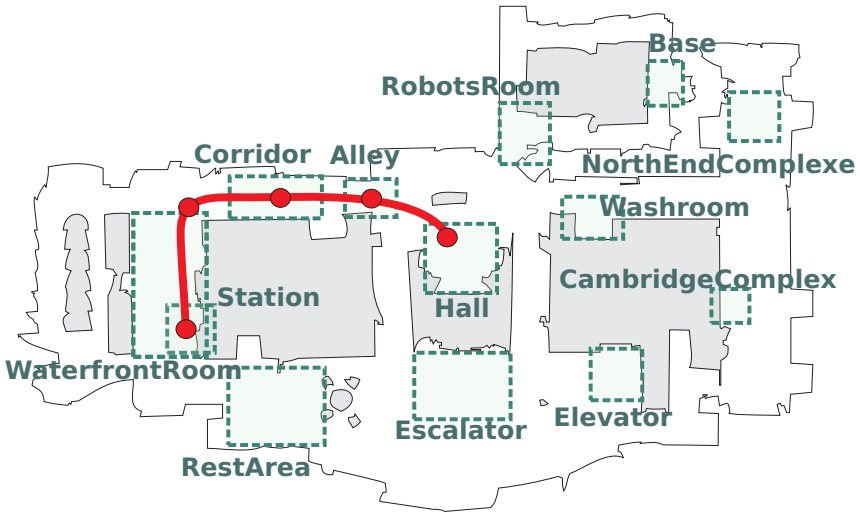

Figure A.7 Our robot's intended trajectory for the technical presentation.

3. PLANNED RECHARGE. Initiate a recharging task, asking somebody to plug the robot in an electrical outlet. This should occur no later than 25 minutes after the beginning of the presentation at the *Station*. However, *Station* is an unknown location when the demonstration starts. There is two ways the location can be determined : from user input with the touch screen interface, or automatically added if an electrical outlet is perceived.

4. PLAYBACK. Accelerated replay of what the robot experienced during the demonstration. The WorkspaceViewer program starts an automatic replay from the beginning of the demonstration, and stops at important events. Judges can then see a snapshot of the active behaviors, the plan, the trajectory of the robot and active tasks. Playback occurs 15 minutes after the beginning of the demonstration at *Station* (max 2 :00 min).

5. QUESTIONS AND COMMENTS. Ask judges to enter comments regarding its performance and capabilities through a WorkspaceViewer contextual interface. Judges could enter textual or vocal comments if they want. This task occurs 20 minutes after the beginning of the presentation (max 4 min). One minute before the end, a 'Thank you' message is displayed on the screen, and our robot gives one business card.

Figure A.8 shows the dry run conducted minutes before the technical presentation, without the jury members. We started our robot in the *Hall* with a mission making it go through *Alley*, *Corridor*, *Waterfront Room*, to finally end its presentation at *Station*. Everything went smootlhy and worked out fine.

Figure A.9 shows what actually occurred during the technical demonstration. Compared to Figure A.8, our robot had a lot of difficulties localizing its position when people and judges were around the robot. CARMEN using laser range finder data had difficulties finding reference points to its internal map. The estimated positions, represented with dots on Figure A.9, are scattered and even sometimes outside the map. Additional fine-

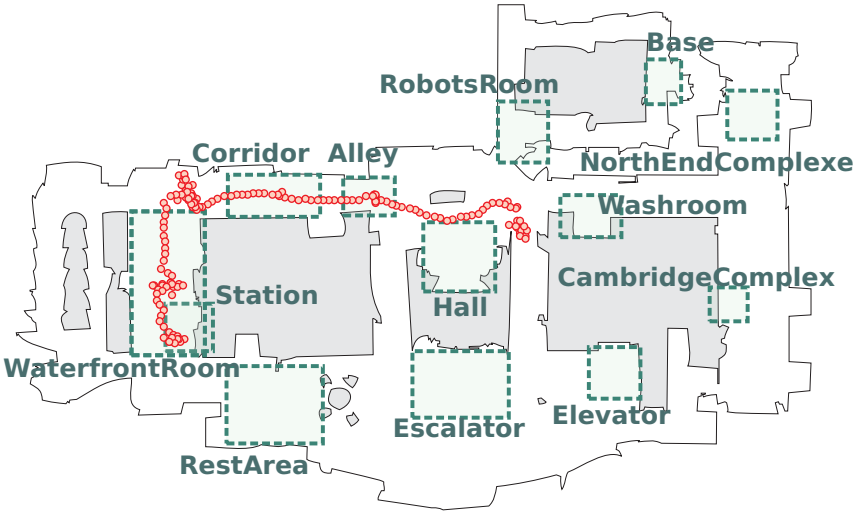

Figure A.8 Localization results with nobody around our robot.

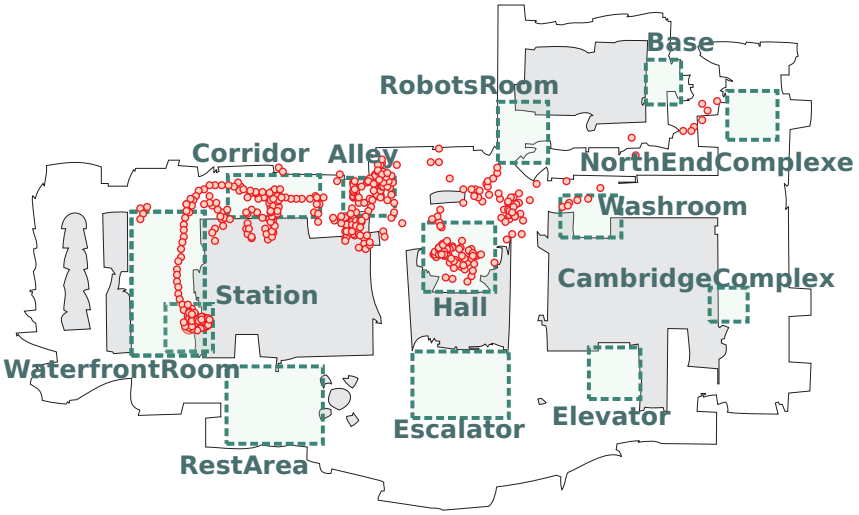

Figure A.9 Localisation results during the technical presentation.

tuning of parameters of probabilistic odometry model in CARMEN configuration file would have been required to improve localization performance in crowded conditions. We had to reposition manually the robot on the map when this happened, and our robot arrived two minutes late at *Alley*. With our robot already following a tight schedule, it had a lot of trouble going through all of the planned tasks : they were dismissed in sequence to try to catch up with the schedule. Most of the time, we had to manually demonstrate the capabilities by modifying the initial plan and removing tasks so that the time constraints could fit in a new plan. This went on until the end, even exceeding the 25 minutes time constraint. Therefore, it turned out that localization performance played a huge impact on the overall performance of our robot. We knew that the time constraints were difficult to meet, but we did not expect such poor performance with localization. This revealed the influence of tightly coupling the planner and the localizer, and that multiple localization capabilities may be required when navigating with too many people surrounding the robot, blocking the laser's field of view to get good position estimates.

Our public demonstration was made to be more interactive and entertaining. For instance, we made our robot interact with people by playing games (music or science trivia questions, fortune cookies). In very crowded conditions, positioning the camera in the direction of the speaker worked very well, as so did the separation of sources considering the very noisy conditions in which the robot was in. Looking at the log files, it is possible to clearly follow conversations the robot had with people. With the event taking place outside the technical demonstration area, no map was made of the area, and our robot only wandered around, being stopped most of the time to interact with people. With the observed performances during the technical demonstration, having a map would not have helped the robot localized itself in the area (it was much more crowded than what our robot experienced during the technical demonstration).

Overall, our robot was the overall winner of the event, finishing first in four categories (Perceptual Learning through Human Teaching ; Perception, Reasoning and Action ; Shared Attention, Common Workspace, Intent Detection ; Integration Challenge), and second in one (Natural Language Understanding and Action Execution). It was also awarded the Public Choice Award.

A.5 Next iteration in designing advanced mobile robots

Our 2006 implementation showed increased reliability and capabilities of MBA (especially the DTW) in a stable and integrated implementation of the components. Our planner is capable of dealing with conditions temporal constraints violation, opportunity detection, task cancellation, unknown waypoint) that would occur in real life settings and within a computational architecture using distributed modules (compared to architectures with one centralized module for goal/task generation). Our robot is equipped with a dynamic viewer with voice and graphical interaction, contextualized based on the robot's active tasks.

However, it is only one step in coming up with the most efficient integration of these modalities. We now have the necessary tools for easy configuration of robot missions and for on- and off-line analysis of experienced situations (trajectories, requests, sound sources, pictures, internal states, etc.), useful in human-robot interaction experiments and AI algorithms in unconstrained conditions. In future work, we want to use these tools to address different research issues, such as the influence of directed attention in vocal interaction, the preferred way and time spent interacting with a robot, the most efficient strategy to navigate in a crowd or to find an unknown location, to schmooze and make acquaintance with people, and to compare different components for the robot's modalities (e.g., speech recognition software such as NUANCE and Sphinx, dialogue software such as CLSU and Collagen, SLAM algorithm such as CARMEN and vSLAM).

Still, we must remember that the intelligence manifested by a machine is limited by its physical, perceptual and reasoning capabilities. Using our robot entries to the AAAI 2000, 2005 and 2006 events, we were able to study the added value of integrating hardware and software components, energetic versus weight versus power considerations, navigation capabilities, artificial audition and vision, task planning and scheduling, distributed decisional architecture and visualization tools for evaluation in dynamic conditions. The software and decisional architectures can be ported and adapted to another platform, but their strengths and limitations can only be adequately characterized when used on platforms with greater capabilities. And we need to add significant improvements to the platform to make even more progress. Therefore, from what we observed from our trials with our robots and in line with the current state-of-the-art in mobile robotic development, we have identified and are working on two key innovations that would greatly benefit our objective of designing advanced mobile robots operating in everyday environments and interacting with humans.

- **Compliance**. A robot mechanically interacts with its environment when the dynamics of the coupled system (the robot and environment in contact) differ significantly from the dynamics of the robot alone [Buerger, 2005; Hogan et Buerger, 2005]. Not all robots mechanically interact with their environment. For instance, the vast majority of robots used for industrial applications do not interact significantly. In many cases, a limited amount of physical interaction can be tolerated without specific modeling or control. However, for complex robotic tasks (e.g., manipulation, locomotion), the lack of knowledge of precise interaction models, the difficulties to precisely measure the task associated physical quantities (e.g., position of contact points, interaction forces) in real-time, the finite sampling time of digital control loops and the non-collocation of sensors and transducers have negative effects on performance and stability of robots when using simple force or simple movement controllers. For robots to take a more significant role in human life, they must safely manage intentional and unintentional physical contact with humans, even while performing high-force tasks. These challenges occur when interacting with the environment produces coupled system dynamics that differ significantly from the dynamics of the robot system alone.

We are currently developing a new compliant actuator named Differential Elastic Actuator (DEA), specifically designed to deal with the lack of knowledge on precise interaction models in natural settings and the difficulties to precisely measure in real-time forces and positions associated to complex robotic tasks like manipulation and locomotion. Compared to FSCA and SEA, DEA [Lauria *et al.*, 2008] uses a differential coupling instead of a serial coupling between a high impedance mechanical speed source and a low impedance mechanical spring. This results in a more compact and simpler solution, with similar performances. DEA are high performance actuators providing higher torque/encumbrance ratio, higher torque/weight ratio, lower rotative inertia, lower rotation velocity of the flexible element, and improved effort transmission through the motor's chassis.

- **Gesture in natural interaction**. Natural interaction is mainly influenced by perception through senses like vision, audition and touch, and by actions like speech, graphics and also gesture. Gesture, i.e., a motion made to express or help express thought, was not covered in our past iterations, and would greatly enhanced the way the robot communicate with humans. Gesture can be made using arms, facial expression, but also through the pose of the robot (rising itself up or moving down).

With these elements in mind, we are currently designing our third iteration of interactive autonomous mobile robot platform. The platform is shown in Figure A.10. The mobile base is a legged-tracked platform capable of changing the orientation of its four articulations. Each articulation has three degrees of freedom (DOF) : it can rotate 360 degrees around its point of attachment to the chassis, can change its orientation over 180 degrees, and rotate to propulse the robot. This platform is inspired from the AZIMUT platform [Michaud *et al.*, 2005b], built using stiff transmission mechanisms, with motors and gearboxes placed directly at the attachment points of the articulations. Such design choices made the platform vulnerable to shocks when moving over rough terrain, especially with its articulations pointing down : an undetected obstacle touching the tip of an articulation would create an important reaction force at the articulation's attachment point to the chassis and inside his non back drivable gearbox. The new base uses DEA actuators for compliant locomotion control : 4 DEA for the direction of its articulations (making it possible to guide the platform simply through touch) ; 4 DEA at the attachment point of the leg-tracks (for force control when the robot stands up). This should improve the locomotion capabilities of the platform, making it omnidirectional and intrinsically safe. The humanoid torso is capable of arm gesture and facial expressions. Its arms will use DEA actuators for safe manipulation of objects and interaction with people. All of these elements are currently being fabricated, assembled and tested, and will be integrated by end of 2008.

A.6 Conclusion

Mobile robotics is one of the greatest domains for systems engineering : it requires the integration of sensors, actuators, energy sources, embedded computing, decision-making algorithms, etc., all into one system. Only technologies and methodologies that work wi-

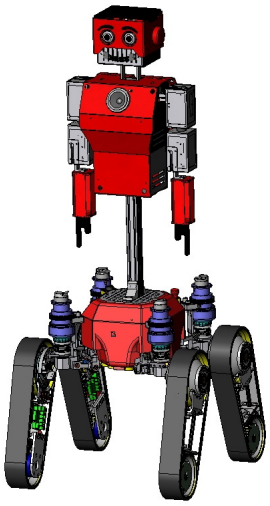

Figure A.10 Our third iteration in designing advanced mobile robotic platform.

thin the interdependent constraints of these elements, addressed in a holistic approach, can be useful. The integration work is influenced by technological factors as well as by the operating conditions of the robotic platform. With continuing technological progress, designs and tools must take into consideration possible extensions to the integration of modular elements of the system. Because a robot is an embodied system, it has to deal with real world dynamics and issues related to the intended application. Therefore, integration directly influences scientific progress in mobile robots, and it is important to address such challenges by developing and addressing them in designing new platforms and conduct real world field trials. The primary research contribution of this paper is to present what we have done so far with our AAAI 2006 entry, and what we are aiming to accomplish with our new design iteration. This puts in perspective what can be done and what remains to be solved.

The design of advanced mobile robotic platforms is still in a basic research phase [Shaw, 2002], investigating ideas and concepts, putting initial structure on problems and framing critical research questions. Progress is accomplished through multiple iterations of increasing complexity, identifying new challenges and research issues as we moved through these iterations. What we have identified with our AAAI 2006 robot entry at all levels, from structural to hardware and software components, has led to the design specifications of our new robot. It is by designing more advanced platforms that we will be able to measure progress at all these levels.

As a corollary, one of our objectives is to share our efforts by making available our hardware and software components. Advanced mobile robotics is a field so broad that it is not possible to be an expert in all related areas. Collaborative work is therefore a necessary element in our attempts, as a community, to significantly accelerate scientific progress of the field. More specifically, we plan to use our new robot to conduct comparative and integrated evaluations of robotic software capabilities and architectures on a common platform and shared test conditions, thus eliminating any bias to conduct such evaluations. This way, we should be able to move research activities from feasibility studies to compa-

rative evaluations and human-robot interaction field studies, addressing an all new level with clear innovations in terms of motion, interaction and cognition capabilities, both individually and integrated.

A.7 Acknowledgment

F. Michaud holds the Canada Research Chair (CRC) in Mobile Robotics and Autonomous Intelligent Systems. Support for this work is provided by the Natural Sciences and Engineering Research Council of Canada, the Canada Research Chair program and the Canadian Foundation for Innovation.

LISTE DES RÉFÉRENCES

Arkin, R. C. (1998). *Behavior-Based Robotics*. The MIT Press.

Asfour, T., Regenstein, K., Azad, P., Schroder, J., Bierbaum, A., Vahrenkamp, N. et Dillman, R. (2006). ARMAR-III : An integrated humanoid platform for sensory-motor control. Dans *Proceedings IEEE/RAS International Conference on Humanoid Robotics*. p. 169–175.

Badali, A., Valin, J.-M., Michaud, F. et Aarabi, P. (2009). Evaluating real-time audio localization algorithms for artificial audition on mobile robots. Dans *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Bayer, S., Doran, C. et George, B. (2001). Exploring speech-enabled dialogue with the Galaxy Communicator infrastructure. Dans *Proceedings of the First International Conference on Human Language Technology Research*. Association for Computational Linguistics, Morristown, NJ, USA, p. 1–3.

Beaudry, E., Brosseau, Y., Côté, C., Raïevsky, C., Létourneau, D., Kabanza, F. et Michaud, F. (2005). Reactive planning in a motivated behavioural architecture. Dans *Proceedings American Association for Artificial Intelligence Conference*. p. 1242–1247.

Bird, S. et Loper, E. (2004). Natural Language Toolkit (NLTK). Dans *Proceedings the Association for Computational Linguistics*. p. 214–217.

Bischoff, R. et Graefe, V. (2004). HERMES : A versatile personal assistant robot. *Proceedings IEEE Special Issue on Human Interactive Robots for Psychological Enrichment*, p. 1759–1779.

Black, A. et Taylor, P. (1997). *Festival Speech Synthesis System : System Documentation (1.1.1)* (Rapport technique). Human Communication Research Centre.

Brachman, R. J. (2006). (AA)AI more than the sum of its parts. *AI Magazine*, volume 27, numéro 4, p. 19–34.

Breazeal, C. (2004). Function meets style : Insights from emotion theory applied to HRI. Dans *IEEE Transactions on Systems, Man, and Cybernetics*. Part C, volume 34-2. p. 187–194.

Bresina, J., Dearden, R., Meuleau, N., Ramkrishnan, S., Smith, D. et Washington, R. (2002). Planning under continuous time and resource uncertainty : A challenge for AI. Dans *Proceedings 19 th Conference on Uncertainty in AI*. p. 77–84.

Briere, S., Valin, J.-M., Michaud, F. et Letourneau, D. (2008). Embedded auditory system for small mobile robots. Dans *Proceedings IEEE International Conference on Robotics and Automation*.

Brock, O., Fagg, A., Grupen, R., Platt, R., Rosenstein, M. et Sweeney, J. (2005). A framework for learning and control in intelligent humanoid robots. *International Journal of Humanoid Robotics*, volume 2, numéro 3, p. 301–336.

Buerger, S. P. (2005). *Stable, High-Force, Low Impedance Robotic Actuators for Human-Interactive Machines*. Thèse de doctorat, MIT.

Caron, A., Charuel, P., Frechette, M., Giraldeau, F., Grenier, F., Griffin, P.-L., Paquin, N. et Vinet, J. (2005). *Rapport final MARIE-U2S* (Rapport technique).

Center for Spoken Language and Understanding (s. d.). *CSLU Toolkit*. `http://cslu.cse.ogi.edu/toolkit/` (page consultée le 21 septembre 2010).

Cole, R., Massaro, D., de Villiers, J., Rundle, B., Shobaki, K., Wouters, J., Cohen, M., Beskow, J., Stone, P., Connors, P., Tarachow, A. et Solcher, D. (1999a). Tools for research and education in speech science. Dans *Proceedings ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education*. p. 45–52.

Cole, R., Massaro, D., Rundle, B., Shobaki, K., Wouters, J., Cohen, M., Beskow, J., Stone, P., Connors, P., Tarachow, A. et Solcher, D. (1999b). New tools for interactive speech and language training : Using animated conversational agents in the classrooms of profoundly deaf children. *M.A.T.I.S.S.E*, volume 1, numéro 1, p. 45–52.

Cote, C., Brosseau, Y., Letourneau, D., Raievsky, C. et Michaud, F. (2006a). Using MARIE in software development and integration for autonomous mobile robotics. *International Journal of Advanced Robotic Systems, Special Issue on Software Development and Integration in Robotics*, volume 3, numéro 1, p. 55–60.

Cote, C., Letourneau, D., Michaud, F., Valin, J.-M., Brosseau, Y., Raievsky, C., Lemay, M. et Tran, V. (2004). Code reusability tools for programming mobile robots. Dans *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. p. 1820–1825.

Cote, C., Letourneau, D., Raievsky, C., Brosseau, Y. et Michaud, F. (2006b). Using MARIE for mobile robot software development and integration. Dans Brugali, D., *Software Engineering for Experimental Robotics*. Springer Tracts on Advanced Robotics.

DeVault, D., Rich, C. et Sidner, C. (2004). Natural language generation and discourse context : Computing distractor sets from the focus stack. Dans *Proceedings 7th International Florida Artificial Intelligence Research Symposium*.

Do, M. et Kambhampati, S. (2003). Sapa : A scalable multi-objective metric temporal planner. *Journal of Artificial Intelligence Research*, volume 20, p. 155–194.

Edsinger-Gonzales, A. et Weber, J. (2004). Domo : A force sensing humanoid robot for manipulation research. Dans *Proceedings IEEE/RAS International Conference on Humanoid Robotics*. p. 273–291.

Fiscus, J. G., Radde, N., Garofolo, J. S., Le, A., Ajot, J. et Laprun, C. (2005). The Rich Transcription 2005 Spring Meeting Recognition Evaluation. Dans *Proceedings*

Multimodal Interaction and Related Machine Learning Algorithms Workshop. p. 369–389.

Gockley, R., Simmons, R., Wang, J., Busquets, D., DiSalvo, C., Caffrey, K., Rosenthal, S., Mink, J., Thomas, S., Adams, W., Lauducci, T., Bugajska, M., Perzanowski, D. et Schultz, A. (2004). *Grace and George : Social robots at AAAI* (Technical Report WS-04-11). AAAI Mobile Robot Competition Workshop.

Hacioglu, K. et Pellom, B. (2003). A distributed architecture for robust automatic speech recognition. Dans *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. p. 328–331.

Hatch, A. O., Peskin, B. et Stolcke, A. (2005). Improved phonetic speaker recognition using lattice decoding. Dans *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. p. 169–172.

Hogan, N. et Buerger, S. P. (2005). Impedance and interaction control. Dans *Robotics and Automation Handbook*. CRC Press.

Huang, X., Alleva, F., Hon, H.-W., Hwang, M.-Y. et Rosenfeld, R. (1993). The SPHINX-II speech recognition system : An overview. *Computer Speech and Language*, volume 7, numéro 2, p. 137–148.

IntRoLab (2010). The ManyEars Project. `http://sourceforge.net/apps/mediawiki/manyears/index.php?title=Main\_Page`.

Lauria, M., Legault, M.-A., Lavoie, M.-A. et Michaud, F. (2008). Differential elastic actuator for robotic interaction tasks. Dans *Proceedings IEEE International Conference on Robotics and Automation*.

Lee, S., Noh, H., Lee, J., Lee, K. et Lee, G. G. (2010). Cognitive effects of robot-assisted language learning on oral skills. Dans *Proceedings INTERSPEECH 2010 Satellite Workshop on Second Language Studies*.

Lesh, N., Marks, J., Rich, C. et Sidner, C. L. (2004). 'Man-computer symbiosis' revisited : Achieving natural communication and collaboration with computers. Dans *IEICE Transactions on Information and Systems*. volume E87-D-6. p. 1290–1298.

Letourneau, D., Michaud, F. et Valin, J.-M. (2004). Autonomous robot that can read. *EURASIP Journal on Applied Signal Processing, Special Issue on Advances in Intelligent Vision Systems : Methods and Applications*, volume 17, p. 1–14.

Lienhart, R. et Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. Dans *Proceedings IEEE International Conference on Image Processing*. volume 1. p. 900–903.

Mathew, B. K., Davis, A. et Fang, Z. (2003). *A Gaussian Probability Accelerator for SPHINX 3* (Rapport technique UUCS-03-02).

Maxwell, B., Smart, W., Jacoff, A., Casper, J., Weiss, B., Scholtz, J., Yanco, H., Micire, M., Stroupe, A., Stormont, D. et Lauwers, T. (2004). 2003 AAAI robot competition and exhibition. *AI Magazine*, volume 25, numéro 2, p. 68–80.

McTear, M. (1999). Software to support research and development of spoken dialogue systems. Dans *Proceedings Eurospeech*. p. 339–342.

Michaud, F. (2002). EMIB - Computational architecture based on emotion and motivation for intentional selection and configuration of behaviour-producing modules. *Cognitive Science Quaterly, Special Issue on Desires, Goals, Intentions, and Values : Computational Architectures*, volume 3-4, p. 340–361.

Michaud, F., Audet, J., Letourneau, D., Lussier, L., Theberge-Turmel, C. et Caron, S. (2001). Experiences with an autonomous robot attending the AAAI Conference. *IEEE Intelligent Systems*, volume 16, numéro 5, p. 23–29.

Michaud, F., Brosseau, Y., Cote, C., Letourneau, D., Moisan, P., Ponchon, A., Raievsky, C., Valin, J.-M., Beaudry, E. et Kabanza, F. (2005a). Modularity and integration in the design of a socially interactive robot. Dans *Proceedings IEEE International Workshop on Robot and Human Interactive Communication*. p. 172–177.

Michaud, F., Cote, C., Letourneau, D., Brosseau, Y., Valin, J.-M., Beaudry, E., Raievsky, C., Ponchon, A., Moisan, P., Lepage, P., Morin, Y., Gagnon, F., Giguere, P., Roux, M.-A., Caron, S., Frenette, P. et F.Kabanza (2007). Spartacus attending the 2005 AAAI Conference. *Autonomous Robots, Special Issue on AAAI Mobile Robot Competition*, volume 22, numéro 4, p. 369–384.

Michaud, F., Letourneau, D., Arsenault, M., Bergeron, Y., Cadrin, R., Gagnon, F., Legault, M.-A., Millette, M., Pare, J.-F., Tremblay, M.-C., Lepage, P., Morin, Y. et Caron, S. (2005b). Multi-modal locomotion robotic platform using leg-track-wheel articulations. *Autonomous Robots, Special Issue on Unconventional Robotic Mobility*, volume 18, numéro 2, p. 137–156.

Michaud, F., Letourneau, D., Beaudry, E., Frechette, M., Kabanza, F. et Lauria, M. (2009). Iterative design of advanced mobile robots. *International Journal of Computing and Information Technology, Special Issue on Advanced Mobile Robotics*, volume 4, p. 1–16.

Michaud, F., Létourneau, D., Fréchette, M., Beaudry, E. et Kabanza, F. (2006). Spartacus, scientific robot reporter. Dans *Proceedings American Association for Artificial Intelligence Conference Workshop*.

Montemerlo, M., Roy, N. et Thrun, S. (2003). Perspectives on standardization in mobile robot programming : The Carnegie Mellon Navigation (CARMEN) toolkit. Dans *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*. p. 2436–2441.

Nakadai, K., Okuno, H. G., Nakajima, H., Hasegawa, Y. et Tsujino, H. (2008). An open source software system for robot audition HARK and its evaluation. Dans *Proceedings IEEE-RAS International Conference on Humanoid Robotics*. p. 561–566.

Nakadai, K., Takahashi, T., Okuno, H., Nakajima, H., Hasegawa, Y. et Tsujino, H. (2010). Design and implementation of robot audition system 'HARK' - Open source software for listening to three simultaneous speakers. *Advanced Robotics*, volume 24, numéro 5-6, p. 739–761.

Nau, D., Au, T., Ilghami, O., Kuter, U., Murdock, J., Wu, D. et Yaman, F. (2003). SHOP2 : An HTN planning system. *Journal of Artificial Intelligence Research*, volume 20, p. 379–404.

Nau, D., Ghallab, M. et Traverso, P. (2004). *Automated Planning : Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Nuttin, M., Vanhooydonck, D., Demeester, E., Brussel, H. V., Buijsse, K., Desimpelaere, L., Ramon, P. et Verschelden, T. (2004). A robotic assistant for ambient intelligent meeting rooms. Dans *Proceedings First European Symposium on Ambient Intelligence*. p. 304–317.

Pellom, B. et Hacioglu, K. (2003). Recent Improvements in the CU SONIC ASR System for Noisy Speech : The SPINE Task. Dans *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. p. 4–7.

Pratt, G. et Williamson, M. (1995). Series elastic actuators. Dans *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*. volume 1. p. 399–406.

Rudnicky, A. et Wu, X. (1999). An agenda-based dialog management architecture for spoken language systems. Dans *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*. p. 3–7.

Rybski, P., Tejada, S., Blank, D., Stroupe, A., Bugajska, M. et Greenwald, L. (2006). The AAAI 2005 Mobile Robot Competition and Exhibition. *AI Magazine*, volume 27, numéro 3.

Scheutz, M., Schermerhorn, P., Kramer, J. et Middendorff, C. (2006). The utility of affect expression in natural language interactions in joint human-robot tasks. Dans *Proceedings IEEE/ACM 1st Annual Conference on Human-Robot Interaction*. p. 226–233.

Shaw, M. (2002). What makes good research in software engineering ? Dans *Proceedings European Joint Conference on Theory and Practice of Software*.

Sidner, C., Kidd, C., Lee, C. et Lesh, N. (2004). Where to look : A study of human-robot engagement. Dans *Proceedings ACM International Conference on Intelligent User Interfaces (IUI)*. p. 78–84.

Simmons, R., Goldberg, D., Goode, A., Montemerlo, M., Roy, N., Sellner, B., Urmson, C., Schultz, A., Abramson, M., Adams, W., Atrash, A., Bugajska, M., Coblenz, M., MacMahon, M., Perzanowski, D., Horswill, I., Zubek, R., Kortenkamp, D., Wolfe, B., Milam, T. et Maxwell, B. (2003). GRACE : An autonomous robot for the AAAI Robot Challenge. *AI Magazine*, volume 24, numéro 2, p. 51–72.

Smart, W. D., Dixon, M., Melchior, N., Tucek, J. et Srinivas, A. (2003). Lewis the graduate student : An entry in the AAAI Robot Challenge. Dans *AAAI Workshop on Mobile Robot Competition*.

Stiehl, W. et Breazeal, C. (2005). Design of a therapeutic robotic companion for relational, affective touch. Dans *Proceedings IEEE Workshop on Robot and Human Interactive Communication*.

Valin, J.-M. (2005). *Auditory System for a Mobile Robot*. Thèse de doctorat, Département de génie électrique et de génie informatique, Université de Sherbrooke, Sherbrooke, Québec, Canada.

Valin, J.-M., Michaud, F. et Rouat, J. (2007a). Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering. *Robotics and Autonomous Systems Journal*, volume 55, numéro 3, p. 216–228.

Valin, J.-M., Rouat, J. et Michaud, F. (2004). Enhanced robot audition based on microphone array source separation with post-filter. Dans *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. p. 2123–2128.

Valin, J.-M., Yamamoto, S., Rouat, J., Michaud, F., Nakadai, K. et Okuno, G. (2007b). Robust recognition of simultaneous speech by a mobile robot. *IEEE Transactions on Robotics*, volume 23, numéro 4, p. 742–752.

Vaughan, R. T., Gerkey, B. P. et Howard, A. (2003). On device abstractions for portable, reusable robot code. Dans *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*. p. 2421–2427.

Walker, M., Y, L. H. et Y, J. A. (2000). Evaluation for DARPA communicator spoken dialogue systems. Dans *Proceedings Second International Conference on Language Resources and Evaluation*.

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P. et Woelfel, J. (2004). *Sphinx-4 : A Flexible Open Source Framework for Speech Recognition* (Rapport technique).

Werner, M. (1999). *Humanism and Beyond the Truth, volume 13*. Humanism Today.

WineHQ support group (s. d.). *Run Windows applications on Linux, BSD, Solaris and Mac OS X)*. `http://www.winehq.org/` (page consultée le 26 septembre 2010).