Spartacus, Scientific Robot Reporter

F. Michaud, D. Létourneau, M. Fréchette, É. Beaudry, F. Kabanza
Université de Sherbrooke
Sherbrooke (Québec) CANADA J1K 2R1
{laborius-challenge}@listes.USherbrooke.ca

Abstract

Spartacus, our 2005 AAAI Mobile Robot Challenge entry, integrated planning and scheduling, sound source localization, tracking and separation, message reading, speech recognition and generation, and autonomous navigation capabilities onboard a custom-made interactive robot. Integration of such a high number of capabilities revealed interesting new issues such as coordinating audio/visual/graphical capabilities, monitoring the impacts of the capabilities in usage by the robot, and inferring the robot's intentions and goals. Our 2006 entry addresses these issues, adding new capabilities to the robot and improving our software and computational architectures, with the objective of increasing and evaluating our understanding of human-robot interaction and integration with an autonomous mobile platform. More specifically, Spartacus is designed to be a scientific robot reporter, in the sense of a human-robot interaction research assistant. The objective is to have Spartacus provide understandable and configurable interaction, intention and information in unconstrained environmental conditions, reporting its experiences for scientific data analysis.

Introduction

In his Presidential Address at the 2005 AAAI Conference, Ronald Brachman argued that Artificial Intelligence (AI) is a system science that must target working in the wild, messy world. 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. Reported issues are 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; Smart *et al.* 2003; Maxwell *et al.* 2004; Simmons *et al.* 2003).

To be useful, these issues must be tackled as a whole. With our 2005 AAAI Mobile Robot Competition entry, our focus was on integrating perceptual and decisional components addressing all four issues, helping us establish new requirements for providing meaningful and appropriate modalities for reporting and monitoring the robot's states and experiences (Michaud *et al.* 2006). Operating in open settings, interactions are fast, diverse and always context-related. Also, having an autonomous robot determining 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.) makes it difficult to understand the robot's behavior just by looking at it. Therefore, we concentrated our integration effort this year to design 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 presents the software/hardware components of the robot, its software and decisional architectures, the technical demonstration made at the conference along with the interfaces developed for reporting the robot's experiences, followed by a discussion regarding the novelties presented and the next challenges to take on to make Spartacus evolve in the future.

Spartacus Software and Hardware Components

Spartacus, shown in Figure 1, is the robotic platform we have designed for high-level interaction with people in real life settings (Michaud *et al.* 2006; 2005). This custom built robot is equipped with a SICK LMS200 laser range finder, a Sony SNC-RZ30N 25X pan-tilt-zoom color camera, a Crossbow IMU400CC-200 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 strategy. Also this year, 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 tech-

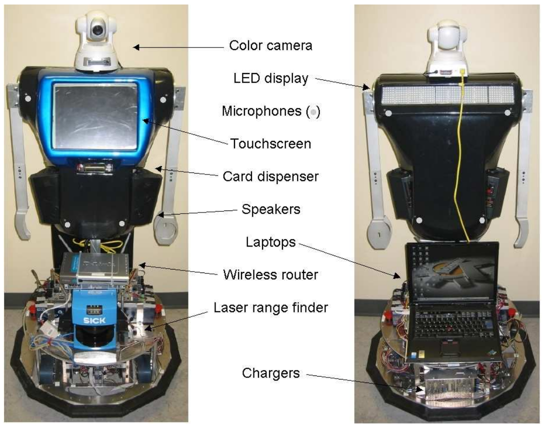


Figure 1: Spartacus (front view, back view).

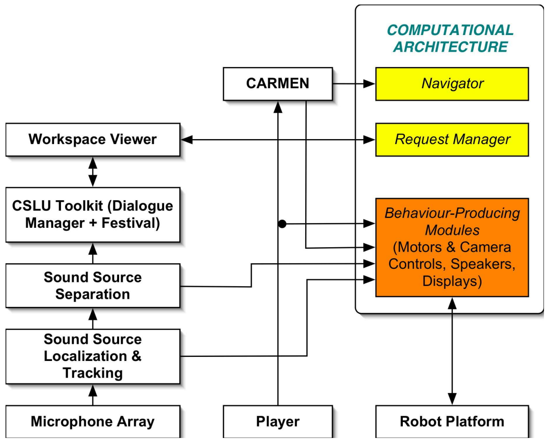

COMPUTATIONAL ARCHITECTURE
CARMEN
Navigator
Workspace Viewer
Request Manager
CSLU Toolkit (Dialogue Manager + Festival)
Behaviour-Producing Modules (Motors & Camera Controls, Speakers, Displays)
Sound Source Separation
Sound Source Localization & Tracking
Microphone Array
Player
Robot Platform

Figure 2: Spartacus software architecture.

nique used is based on a cascade of boosted classifiers working with Haar-like features (Lienhart & Maydt 2002)[1] and a rule-based analysis of the images.

Figure 2 illustrates Spartacus' software architecture. It integrates Player for sensor and actuator abstraction layer (Vaughan, Gerkey, & Howard 2003), CARMEN (Carnegie Mellon Robot Navigation Toolkit) for path planning and localization (Montemerlo, Roy, & Thrun 2003). Also integrated but not shown on the figure is Stage/Gazebo for 2D and 3D simulators, and Pmap library[2] 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, Michaud, & Valin 2004) and SSLTS, our real-time sound source localization, tracking (Valin, Michaud, & Rouat 2006) and separation (Valin, Rouat, & Michaud 2004) system. For speech recognition and dialogue management, we interfaced this year the CSLU toolkit[3]. 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.* 2006b; 2006a).

Decisional Architecture and Modules

For our participation to the AAAI 2006 Mobile Robot Competition, Spartacus is designed to be a scientific robot reporter, in the sense of a human-robot interaction research assistant. The robot is programmed to respond to requests from people, and with the only 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 ask to be plugged in). Spartacus' duty is to address these requests by doing the best with 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 particular time and place, to schmooze, etc. The robot may receive multiple requests at different periods, and will have to autonomously manage what, when and how it will satisfy them.

With Spartacus, 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. 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 we are developing for Spartacus' decision-making capabilities is shown in Figure 3. It is based on the notion of motivated selection of behavior-producing modules. We refer to it as MBA, for Motivated Behavioral Architecture (Michaud *et al.* 2006; 2005). 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 BPM Arbitration and the BPM's activation conditions, as derived by the BPM Selection module.
2. Motivational sources (or Motivations, serving to propel an agent in a certain direction) 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 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.

[1]http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/ref/OpenCVRef_Experimental.htm
[2]http://robotics.usc.edu/ ahoward/pmap/
[3]http://cslu.cse.ogi.edu/toolkit/

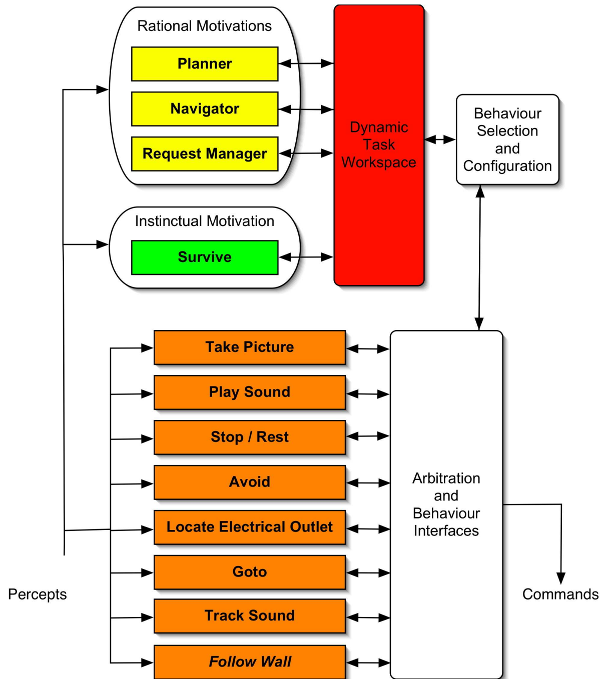


Figure 3: Spartacus decisional architecture.

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, but at a higher abstraction level.

3. Dynamic Task Workspace (DTW) serve 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, queries or subscribe to events regarding the task's status.

It is by exchanging information through the DTW that 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 it perceives a charging station.

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 high-level tasks under temporal constraints and the robot's capabilities (as defined by BPMs). This motivational source is detailed in the following subsection. *Navigator* determines the path to a specific location according to 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 the 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 take 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 & 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 state (this is done by checking its precondition and updating the current 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 blow up 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; Nau, Ghallab, & Traverso 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) allowing to specify 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 much lower than the real average speed 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 & 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 is would be acceptable depending on the environment conditions.

We implement a robot mission as list of task 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; and so on, 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 & Kambhampati 2003), our planner has an anytime planning capability, which are 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.

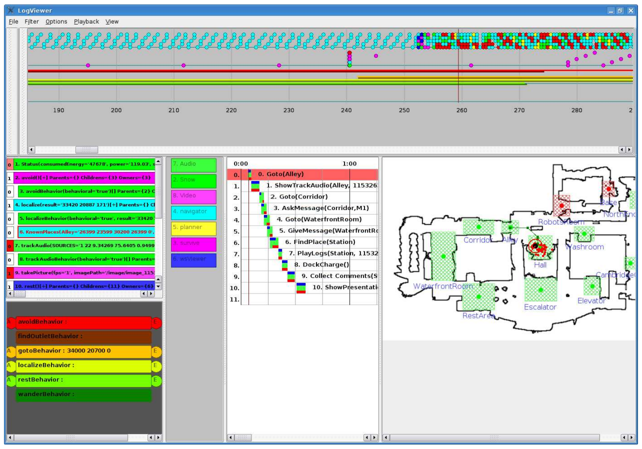
On-line Task Planning The integration of planning capabilities into a robotic system having must take into account sharp real-time constraints and unexpected external events that conflict with previously generated plans. To reduce processing delays in the system, our planner is implemented as a library. 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 validating that each action is executed with the time intervals set in the plan. Violating conditions are handled by specific behavioral modules including one that 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 the navigator motivational source (using CARMEN path planner) and made available through the DTW. Therefore, *Planner* only has to deal with high-level navigation tasks, and it is not necessary to plan intermediate waypoints between two waypoint 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

Here is the set of modalities designed to improve our understanding of integrated modalities (vision, audition, graphical, navigation):

- We extended the capabilities of the LogViewer application, developed in 2005 to monitor off-line the complete set of states of the robot (through log files created by the different software components). We now have an on-line version of the LogViewer to provide a dynamic and real-time view of what the robot is actually doing and planning to do. We call this updated version the WorkspaceViewer. With the WorkspaceViewer, it is now possible to display, directly on Spartacus' 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 LogViewer application to replay logs off-line. Figure 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), 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 first window). The WorkspaceViewer is directly connected to the DTW and displays information as they become available in real-time. The WorkspaceViewer is linked with the Request Manager motivational to handle user requests.



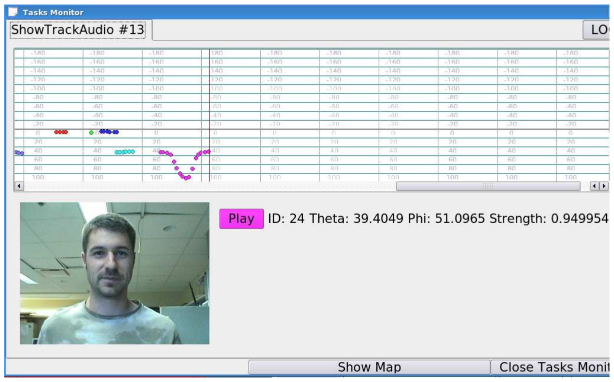
```
0 1. Status(consumedEnergy='47678', power='119.03', s
1 2. avoid()[+] Parents={} Childrens={3} Owners={3}
0 3. avoidBehavior(behavioral='true')[] Parents={2} C
1 4. localize(result='33420 20887 171')[+] Parents={} Cl
0 5. localizeBehavior(behavioral='true', result='33420
0 6. KnownPlaces(Alley='26399 23599 30200 26399 0',
0 7. trackAudio(SOURCES='1 22 9.34269 75.6405 0.9499
0 8. trackAudioBehavior(behavioral='true')[] Parents=
1 9. takePicture(fps='1', imagePath='/image/image_115
1 10. rest()[+] Parents={} Childrens={11} Owners={6}
```

```
A avoidBehavior : E
  findOutletBehavior :
A gotoBehavior : 34000 20700 0 E
A localizeBehavior : E
A restBehavior : E
  wanderBehavior :
```

```
7. Audio
2. Snow
8. Video
4. navigator
5. planner
3. survive
6. wsViewer
```

```
     0:00                  1:00
0.   0. Goto(Alley)
1.   1. ShowTrackAudio(Alley, 115326
2.   2. Goto(Corridor)
3.   3. AskMessage(Corridor,M1)
4.   4. Goto(WaterfrontRoom)
5.   5. GiveMessage(WaterfrontR
6.   6. FindPlace(Station)
7.   7. PlayLogs(Station, 11532
8.   8. DockCharge()
9.   9. Collect Comments(S
10.  10. ShowPresentati
11.
```

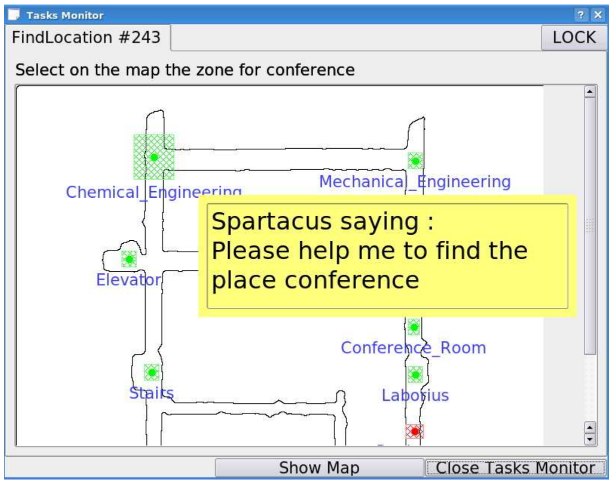
Figure 4: WorkspaceViewer's main window.



ID: 24 Theta: 39.4049 Phi: 51.0965 Strength: 0.949954

Figure 5: Track audio interface.

- Visualization of the SSLTS results. Figure 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 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 block (allowing to diagnose potential difficulties that the speech recognition module has to face). More than 6600 audio streams were recorded over the twelve hours of operation of the robot at the AAAI 2006 conference, held at Seaport Hotel and World Trade Center in Boston. By constructing such database of audio streams, we will be able to evaluate in a more diverse set of conditions the performance of SSLTS and speech recognition algorithms.
- Develop contextual interfaces and interaction capabilities according to the state and intentions of the robot. It is sometimes difficult to know exactly which task is prioritized by the robot at a given time, making the users un-



Spartacus saying :
Please help me to find the place conference

Figure 6: Graphical display for an assistance request to find a location.

able 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 6 illustrates the graphical interface with which Spartacus 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 Spartacus to do. We also made it possible for the users to make these requests verbally, saying outload the names of the buttons. To facilitate the recognition process, a specific grammar and dialogue manager are loaded in CSLU for each graphical mode. The audio and graphical interaction are therefore tightly integrated together.

Demonstration and Results

Spartacus demonstrated its capabilities in the Human-Robot Interaction Event, evaluated in a technical and a public demonstrations. From the seven possible categories in this event, Spartacus participated to five 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.
- 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.

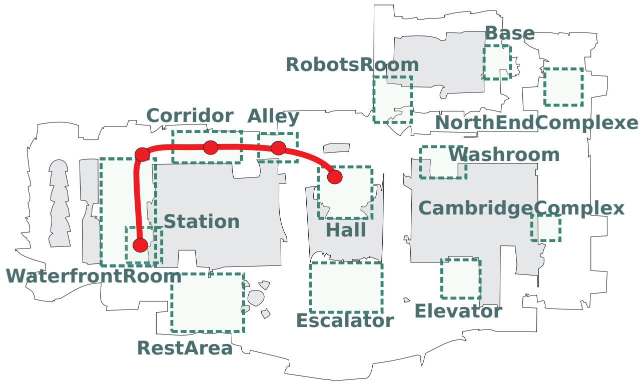


Figure 7: Spartacus' intended trajectory for the technical presentation

- 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, consisted of five phases done in the area represented in Figure 7:

1. INTRODUCTION. Presentation (max 4:30 min) at Alley of Spartacus' 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).
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 the Station location (max 2:00 min).
5. QUESTIONS AND COMMENTS. Ask judges to enter comments regarding its performance and capabilities

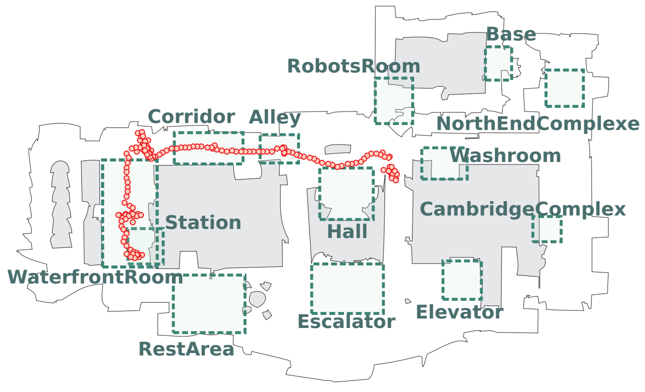


Figure 8: Localization results with nobody around Spartacus

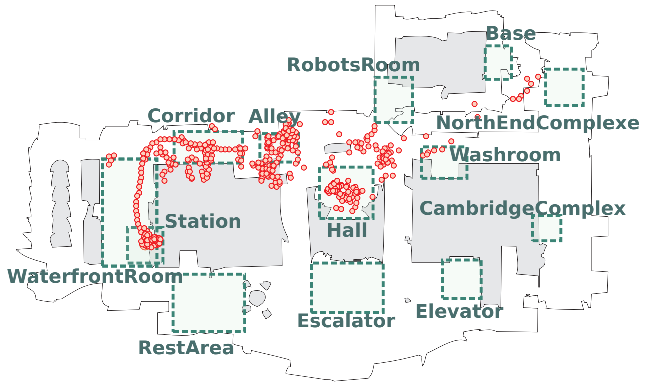


Figure 9: Localisation results during the technical presentation

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 Spartacus gives one business card.

Figure 8 shows the trial run conducted minutes before the technical presentation, without the jury members. We started Spartacus' 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 9 shows what actually occurred during the technical demonstration. Compared to Figure 8, Spartacus 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 9, are scattered and even sometimes outside the map. We had to reposition manually the robot on the map when this happened, and Spartacus arrived two minutes late at Alley. With Spartacus 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 to 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 Spartacus. We knew that the time constraints were difficult to meet, but we did not expect such poor performance with localization. It revealed the influence of tightly coupling the planner and the localizer, and that multi-modal localization capabilities are 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 Spartacus 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. With the event taking place outside the technical demonstration area, no map was made of the area, and Spartacus 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 Spartacus experienced during the technical demonstration).

Conclusion

Overall, our 2006 implementation showed increased reliability and capabilities of MBA (especially the DTW) in a stable and integrated implementation of the components described in the paper. Extending the LogViewer concept, Spartacus is now equipped with a dynamic viewer with voice and graphical interaction, contextualized based on the robot's active tasks. 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). However, it is only a first 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).

Acknowledgments

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. The authors also want to acknowledge the work of A. Caron, P. Charuel, M. Fréchette, F. Giraldeau, F. Grenier, P.-L. Griffin, N. Paquin and J. Vinet for their work on the electric outlet recognition component.

References

Bresina, J.; Dearden, R.; Meuleau, N.; Ramkrishnan, S.; Smith, D.; and Washington, R. 2002. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proc. 19th Conf. on Uncertainty in AI*, 77–84.

Cote, C.; Letourneau, D.; Michaud, F.; Valin, J.-M.; Brosseau, Y.; Raievsky, C.; Lemay, M.; and Tran, V. 2004. Code reusability tools for programming mobile robots. In *Proc. IEEE/RSJ Int. Conf. Intel. Robots & Sys.*, 1820–1825.

Cote, C.; Brosseau, Y.; Letourneau, D.; Raievsky, C.; and Michaud, F. 2006a. Using MARIE in software development and integration for autonomous mobile robotics. *Int. J. of Advanced Robotic Systems* 3(1):55–60.

Cote, C.; Letourneau, D.; Raievsky, C.; Brosseau, Y.; and Michaud, F. 2006b. Using MARIE for mobile robot software development and integration. In Brugali, D., ed., *Software Engineering for Experimental Robotics*. Springer.

Do, M., and Kambhampati, S. 2003. Sapa: A scalable multi-objective metric temporal planner. *J. Artificial Intelligence Research* 20:155–194.

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.; and Schultz, A. 2004. Grace and George: Social robots at AAAI. Technical Report WS-04-11, AAAI Mobile Robot Competition Workshop. pp. 15-20.

Letourneau, D.; Michaud, F.; and Valin, J.-M. 2004. Autonomous robot that can read. *EURASIP Journal on Applied Signal Processing* 17:1–14.

Lienhart, R., and Maydt, J. 2002. An extended set of Haar-like features for rapid object detection. In *Proc. IEEE Int. Conf. Image Processing*, volume 1, 900–903.

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

Michaud, F.; Brosseau, Y.; Cote, C.; Letourneau, D.; Moisan, P.; Ponchon, A.; Raievsky, C.; Valin, J.-M.; Beaudry, E.; and Kabanza, F. 2005. Modularity and integration in the design of a socially interactive robot. In *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication*, 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.; and F.Kabanza. 2006. Spartacus attending the 2005 AAAI conference. *Autonomous Robots*. to appear.

Montemerlo, M.; Roy, N.; and Thrun, S. 2003. Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (CARMEN) toolkit. In *Proc. IEEE/RSJ Int. Conf. Intel. Robots & Sys.*, 2436–2441.

Nau, D.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, J.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *J. Artificial Intelligence Research* 20:379–404.

Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publ.

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.; and Maxwell, B. 2003. Grace : An autonomous robot for the AAAI robot challenge. *AI Magazine* 24(2):51–72.

Smart, W. D.; Dixon, M.; Melchior, N.; Tucek, J.; and Srinivas, A. 2003. Lewis the graduate student: An entry in the AAAI robot challenge. Technical report, AAAI Workshop on Mobile Robot Competition. p. 46-51.

Valin, J.-M.; Michaud, F.; and Rouat, J. 2006. Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering. *Robotics and Autonomous Systems J.*. to appear.

Valin, J.-M.; Rouat, J.; and Michaud, F. 2004. Enhanced robot audition based on microphone array source separation with post-filter. In *Proc. IEEE/RSJ Int. Conf. Intel. Robots & Sys.*, 2123–2128.

Vaughan, R. T.; Gerkey, B. P.; and Howard, A. 2003. On device abstractions for portable, reusable robot code. In *Proc. IEEE/RSJ Int. Conf. Intel. Robots & Sys.*, 2421–2427.

Werner, M. 1999. *Humanism and beyond the truth*, volume 13. Humanism Today. http://www.humanismtoday.org/vol13/werner.html.