A Mixed-Initiative Approach to Human-Robot Interaction in Rescue Scenarios

Alberto Finzi
Dipartimento di Informatica e Sistemistica
Università degli Studi di Roma "La Sapienza"
Via Salaria 113, 00198 Roma Italy
finzi@dis.uniroma1.it

Andrea Orlandini
Dipartimento di Informatica e Automazione
Università degli Studi di Roma TRE
Via della Vasca Navale 79, 00146 Roma Italy
orlandin@dia.uniroma3.it

Abstract

In this paper we present a mixed-initiative planning approach to human-robot interaction in a rescue domain. We deploy a model-based executive monitoring system to coordinate the operator's interventions and the concurrent activities of a rescue rover. We show that this approach can enhance both operator situation awareness and human-robot interaction for the execution and control of the diverse activities needed in rescue missions. We have implemented this control architecture on a robotic system (DORO) and tested it in rescue arenas comparing its performances in different settings.

Introduction

Urban search and rescue (USAR) deals with response capabilities for facing urban emergencies, and it involves the location and rescue of people trapped because of a structural collapse. Starting in 2000, the National Institute of Standard Technology (NIST) together with the Japan National Special Project for Earthquake Disaster Mitigation in Urban Areas (Tadokoro *et al.* 2000; Tadokoro 2000; Maxwell *et al.* 2004; Jacoff, Messina, & Evans 2001) has initiated the USAR robot competitions. NIST, in particular, features future standards of robotics infrastructures, pioneering robotics participation to rescue missions. RoboCup Rescue contests are a test-bed of the technology development of NIST project, and are becoming a central international event for rescue robots, and a real challenge for the robotics community. Rescue robots uphold human operators exploring dangerous and hazardous environments and searching for survivors.

A crucial aspect of rescue environments, discussed in (Burke *et al.* 2004) and (Murphy 2004) concerns the operator situation awareness and human-robot interaction. In (Murphy 2004) the difficulties in forming a mental model of the "robot eye" are endorsed, pointing out the role of the team. Differently from real tests, like the one in Miami (see (Burke *et al.* 2004)), during rescue competitions the operator is forced to be alone while coordinating the robot activities, since any additional team member supporting the operator would penalize the mission. The operator can follow the robot activities only through the robot perception of the environment, and its internal states. In this sense, the overall control framework has to capture the operator attention towards "what is important" so as to make the correct choices: follow a path, enter a covert way, turn around an unvisited corner, check whether a visible victim is really reachable, according to some specific knowledge acquired during the exploration. In this setting, a fully manual control over a robot rescue is not effective (Bruemmer *et al.* 2003): the operator attention has to be focused over a wide range of activities, losing concentration on the real rescue mission objective, i.e. locating victims. Moreover, a significant level of training is needed to teleoperate a rescue rover. On the other hand, fully autonomous control systems are not feasible in a rescue domain where too many capabilities are needed. Therefore, the integration of autonomous and teleoperated activities is a central issue in rescue scenarios and has been widely investigated (Kiesler & Hinds 2004; Yanco & Drury 2002; Drury, Scholtz, & Yanco 2003; Michael Baker & Yanco 2004; Yanco & Drury 2002).

In this work we describe a mixed-initiative planning approach (Ai-Chang *et al.* 2004; Myers *et al.* 2003; Allen & Ferguson 2002; Burstein & McDermott 1996) to Human-Robot Interaction (HRI) in a rescue domain and illustrate the main functionalities of a rescue robot system[1]. We deploy a model-based executive monitoring system to interface the operators' activities and the concurrent functional processes in a rescue rover. In this setting, the user's and the robot's activities are coordinated by a continuos reactive planning process which has to (i) check the execution status with respect to a declarative model of the system; (ii) provide proactive activity while mediating among conflicting initiatives. In particular, we show that this approach can enhance both the operator situation awareness and human-robot interaction for the execution and control of the diverse activities needed during a complex mission such as the rescue one.

The advantage of this approach can be appreciated considering the HRI awareness discussed in (Drury, Scholtz, & Yanco 2003):

- robot-human interaction: given a declarative model of the robot activities, the monitoring system can be "self-aware" about the current situation, at different levels of

[1] Doro is the third award winner in Lisbon contest (2004)

Figure 1: The mobile robot DORO, in a yellow arena.

abstraction; in this way, complex and not nominal interactions among activities can be detected and displayed to the operator;

- human-robot interaction: the operator can take advantage of basic functionalities like mapping, localization, learning vantage points for good observation, victim detection, and victim localization; these functionalities purposely draw his attention toward the current state of exploration, while he interacts with a mixed initiative reactive planner (Ai-Chang *et al.* 2004).

Finally, the humans' overall mission can take advantage of the model, that keeps track of the robot/operator execution history, goals, and subgoals. Indeed, the proposed control system provides the operator with a better perception of the mission status.

Rescue Scenario

NIST has developed physical test scenarios for rescue competitions. There are three NIST arenas, called yellow, orange, and red, of varying degrees of difficulty. A yellow arena represents an indoor flat environment with minor structural damage (e.g. overturned furniture), an orange arena is multilevel and has more rubble (e.g. bricks), a red one represents a very damaged unstructered environment: multilevel, large holes, rubber tubing etc. The arenas are accessible only by mobile robots controlled by one or more operators from a separated place. The main task is to locate as many victims as possible in the whole arena.

Urban search and rescue arena competitions are very hard test-beds for robots and their architectures. In fact, the operator-robot has to coordinate several activities: exploring and mapping the environment, avoiding obstacles (bumping is severely penalized), localizing itself, searching for victims, correctly locating them on the map, identifying them through a numbered tag, and finally describing their own status and conditions.

For each mission there is a time limit of 20 minutes, to simulate the time pressure in a real rescue environment. In this contest human-robot interaction has a direct impact on the effectiveness of the rescue team performance.

We consider the NIST yellow arena as the test-bed for our control architecture. It is mounted on our robotic platform (DORO) whose main modules are: *Map*, managing the algorithm of map construction and localization; *Navigation*, guiding the robot through the arena with exploration behaviour and obstacle's avoidance procedures; *Vision*, used in order to automatically locate victims around the arena.

In this context, (Murphy 2004) propose a high level sequence tasks cycle as a reference for the rescue system behaviour: Localize, Observe general surroundings, look specially for Victims, Report (LOVR). Our interpretation of the cycle corresponds to the following tasks sequence: map construction, visual observation, vision process execution and victim's presence report.

Human Robot Interaction and Mixed Initiative Planning in Rescue Arenas

There have been several efforts to establish the essential aspects of human-robot interaction, given the current findings and state of the art concerning robot autonomy and its modal-abilities towards humans and environments (see e.g.(Dautenhahn & Werry 2000; Kiesler & Hinds 2004; Burke *et al.* 2004; Sidner & Dzikovska 2002; Lang *et al.* 2003) and the already cited (Murphy 2004; Michael Baker & Yanco 2004; Yanco & Drury 2002; Drury, Scholtz, & Yanco 2003), specifically related to the rescue environment. It is therefore crucial to model the interaction in terms of a suitable interplay between supervised autonomy (the operator is part of the loop, and decides navigation strategies according to an autonomously drawn map, and autonomous localization, where obstacle avoidance is guaranteed by the robot sensory system) and full autonomy (e.g. visual information is not reliable because of darkness or smoke etc., and the operator has to lean upon the robot exploration choices).

In order to allow the tight interaction described above, we designed a control system where the HRI is fully based on a mixed-initiative planning activity. The planning process is to continuously coordinate, integrate, and monitor the operator interventions and decisions with respect to the ongoing functional activities, taking into account the overall mission goals and constraints. More precisely, we developed an interactive control system which combines the following features:

- **Model-based control.** The control system is endowed with declarative models of the controllable activities, where causal and temporal relations are explicitly represented (Muscettola *et al.* 2002; Williams *et al.* 2003; Muscettola *et al.* 1998). In this way, hard and soft constraints can be directly encoded and monitored. Furthermore, formal methods and reasoning engines can be deployed either off-line and on-line, to check for consistency, monitor the executions, perform planning or diagnosis. In a mixed-initiative setting the aim of a model-based system is twofold: on the one hand the operator ac-

tivities are explicitly modeled and supervised by the control system; on the other hand, the model-based monitoring activity exports a view of the system that is intuitive and readable by humans, hence the operator can further supervise the robot status in a suitable human robot interface.

- **Reactive executive monitoring.** Given this model, a reactive planning engine can monitor both the system's low-level status and the operator's interventions by continuously performing sense-plan-act cycles. At each cycle the reactive planner has to: (i) monitor the consistency of the robot and operator activities (w.r.t. the model) managing failures; (ii) generate the robot's activities up to a planning horizon. The short-range planning activity can also balance reactivity and goal-oriented behaviour: short-term goals/tasks and external/internal events can be combined while the planner tries to solve conflicts. In this way, the human operator can interact with the control system through the planner in a mixed initiative manner.
- **Flexible interval planning.** At each execution cycle a flexible temporal plan is generated. Given the domain uncertainty and dynamics, time and resources cannot be rigidly scheduled. On the contrary, it is necessary to account for flexible behaviours, allowing one to manage dynamic change of time and resource allocation at execution time. For this reason the start and end time of each scheduled activity is not fixed, but the values span a temporal interval.
- **High-level agent programming.** The high-level agent programming paradigm allows one to integrate procedural programming and reasoning mechanisms in a uniform way. In this approach, the domain's first principles are explicitly represented in a declarative relational model, while control knowledge is encoded by abstract and partial procedures. Both the system's and the operator's procedural operations can be expressed by high-level partial programs which can be completed and adapted to the execution context by a program interpreter endowed with inference engines.

Control Architecture

In this section, we describe the control system we have defined to incorporate the design principles introduced above. Following the approach in (Muscettola *et al.* 2002; Williams *et al.* 2003; Volpe *et al.* 2001; Finzi, Ingrand, & Muscettola 2004) we introduce a control system where decision processes (including declarative activities and operator's interventions) are tightly coupled with functional processes through a model-based executive engine. Figure 2 illustrates the overall control architecture designed for DORO. The physical layer devices are controlled by three functional modules associated to the main robots activities (mapping and localization, visual processing, and navigation). The *state manager* and *task dispatcher* in the figure are designed to manage communication between the executive and functional layers.

The *state manager* gets from each single module its current status so that any module can query the state manager about

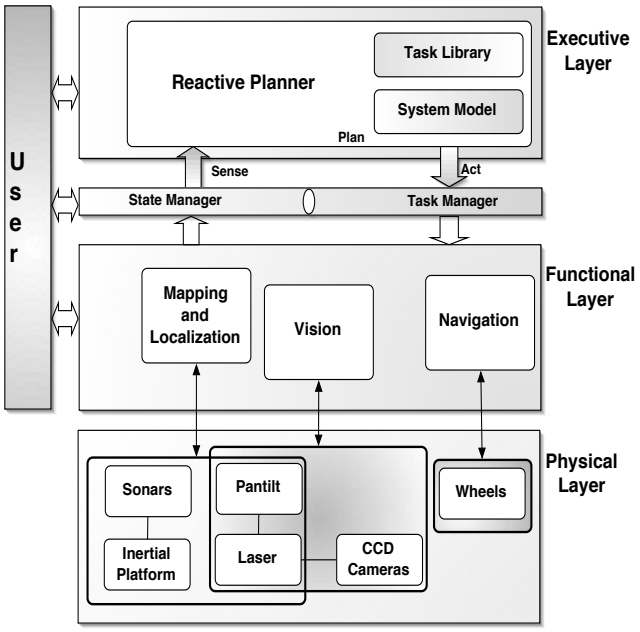

Figure 2: Control architecture

the status of any another module. The state manager updates its information every 200 msec., the task dispatcher sends tasks activation signals to the modules (e.g. map_start) upon receiving requests from the planner or the human operator. The overall computational cycle works as follows: the planner gets the modules status querying the state manager. Once the state manager provides the execution context, the planner produces a plan of actions (planning phase about 0.5 sec.) and yields the first set of commands to the task dispatcher. In the execution phase (about 0.5 sec.), each module reads the signals and starts its task modifying its state. At the next cycle start, the planner reads the updated status through the state manager and can check whether the tasks were correctly delivered. If the status is not updated as expected, a failure is detected, the current plan is aborted and a suitable recovery procedure is called.

Functional Modules. As mentioned above, the functional layer is endowed with three main modules: *Mapping and Localization*, *Navigation*, and *Vision*. These modules provide different tasks that can be activated or stopped according to the *start* or *end* actions communicated by the task dispatcher.

A functional module is a reactive component that changes its internal status with respect to the action received from the task dispatcher. Nevertheless, it can also provide some proactiveness, by suggesting the planner/operator an action to be executed. For instance, the *Slam* module assumes a particular mode in order to communicate to the system that a map's construction cycle is ended, and then the control system can decide an action to stop the mapping phase. Morover, some modules can directly interact among them by communicating some low-level information bypassing the

state manager (and the executive layer), e.g. *Slam* devises to *Navigation* the coordinates of the nearest unexplored point during the exploration phases.

User interaction. The human operator can interact with the control loop both during the plan and the act phase. In the planning phase, the operator can interact with the control system by: (i) posting some goals which are to be integrated in the partial plan already generated; (ii) modifying the generated plan through the user interface; (iii) on-line changing some planning parameters, like the planning horizon, the lenght of the planning cycle, etc.. In the executive phase, the user can directly control some functional modules (e.g., deciding where the rover is to go, or when some activities are to stop). In this case, the human actions are assimilated to exogenous events the monitoring system is to manage and check. Finally, the operator's actions can be accessed by the state manager, and, analogously to the functional modules, can be monitored by the model-based control system.

Model-Based Monitoring

The role of a model-based monitoring system is to enhance both the system safeness and the operator situation awareness. Given a declarative representation of the system causal and temporal properties, the flexible executive control is provided by a reactive planning engine which harmonizes the operator activity (commands, tasks, etc.) with the mission goals and the reactive activity of the functional modules. Since the execution state of the robot is continuously compared with a declarative model of the system, all the main parallel activities are integrated into a global view and subtle resources and time constraints violations can be detected. In this case the planner can also start or suggest recovery procedures the operator can modify, neglect, or respect. Such features are implemented by deploying *high-level agent programming* in Temporal Concurrent Golog (Reiter 2001; Pirri & Reiter 2000; Finzi & Pirri 2004) which provides both a declarative language (i.e. Temporal Concurrent Situation Calculus (Pinto & Reiter 1995; Reiter 1996; Pirri & Reiter 2000)) to represent the system properties and the planning engine to generate control sequences.

Temporal Concurrent Situation Calculus. The Situation Calculus (SC) (McCarthy 1963) is a sorted first-order language representing dynamic domains by means of *actions*, *situations*, i.e. sequences of actions, and *fluents*, i.e. situation dependent properties. Temporal Concurrent Situation Calculus (TCSC) extends the SC with time and concurrent actions. In this framework, concurrent durative processes (Pinto & Reiter 1995; Reiter 1996; Pirri & Reiter 2000) can be represented by fluent properties started and ended by durationless actions. For example, the process $going(p_1, p_2)$ is started by the action $startGo(p_1, t)$ and it is ended by $endGo(p_2, t')$.

Declarative Model in TCSC. The main processes and states of DORO are explicitly represented by a declarative dynamic-temporal model specified in the Temporal Concurrent Situation Calculus (TCSC) . This model represents cause-effect relationships and temporal constraints among the activities: the system is modeled as a set of *components* whose state changes over time. Each component (including the operator's operations) is a concurrent thread, describing its history over time as a sequence of states and activities. For example, in the rescue domain some components are: *pant-tilt*, *slam*, *navigation*, *visualPerception*, etc.

Each of these is associated with a set of processes, for instance some of those are the following: *SLAM* can perform $slmMap$ to map the environment and $slmScan$ to acquire laser measures; *visualPerception* can use $visProcess(x)$ to process an image x. *navigation* can explore a new area ($nvWand$) or reach a target point x ($nvGoTo$); *pan-tilt* can deploy $ptPoint(x)$ (moving toward x) and $ptScan(x)$ (scanning x). The history of states for a component over a period of time is a *timeline*. Figure 3 illustrates a possible evolution of $navigation$, $slam$, and $pan\text{-}tilt$ up to a planning horizon.

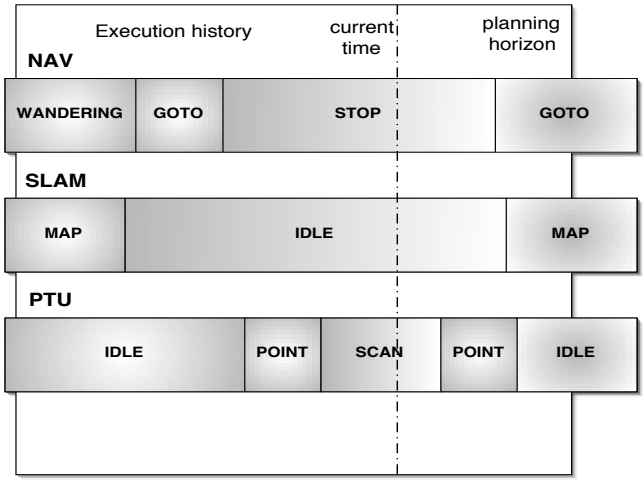

Figure 3: Timelines evolution

Hard time constraints among activities can be defined by a temporal model using Allen-like temporal relations, e.g.: $ptPoint(x)$ *precedes* $ptScan(x)$, $ptScan(x)$ *during* $nvStop$, etc..

Temporal Concurrent Golog. Golog is a situation calculus-based programming language which allows one to define procedural scripts composed of primitive actions explicitly represented in a SC action theory. This hybrid framework integrates procedural programming and reasoning about the domain properties. Golog programs are defined by means of standard (and not so-standard) Algol-like control constructs: (i) action sequence: $p_1; p_2$, (ii) test: $\phi?$, (iii) nondeterministic action choice $p_1|p_2$, (iv) conditionals, while loops, and procedure calls. Temporal Concurrent Golog (TCGolog) is the Golog version suitable for durative and parallel actions, it is based on TCSC and allows parallel

action execution: $a \| b$. An example of a TCGolog procedure is:

$$\begin{array}{l}\mathbf{proc}(observe(x),\\ \ \mathbf{while}\ (nvStop \wedge \neg obs(x))\ \mathbf{do}\ \pi(t_1, start(t_1)? :\\ \ \ [\mathbf{if}\ (ptIdle(0))\ \mathbf{do}\ \pi(t_2, startPoint(x, t_1) : (t_2 - t_1{<}3)?)|\\ \ \ \ \mathbf{if}\ (ptIdle(x))\ \mathbf{do}\ \pi(t_3, startScan(x, t_3) : (t_3 - t_1{<}5)?))).\end{array}$$

Here the nondeterministic choice between $startPoint$ and $startScan$ is left to the Golog interpreter which has to decide depending on the execution context. Note that, time constraints can be encoded within the procedure itself. In this case the procedure definition leaves few nondetermistic choices to the interpreter. More generally, a Golog script can range from a completely defined procedural program to an abstract general purpose planning algorithm like the following:

$$\begin{array}{l}\mathbf{proc}(plan(n),\\ \quad true? \mid \pi(a, (primitive\_action(a))? \ :\ a)\ :\ plan(n-1))\end{array}$$

The semantics of a Golog program δ is a situation calculus formula $Do(\delta, s, s')$ meaning that s' is a possible situation reached by δ once executed from the situation s. For example, the meaning of the $a|b$ execution is captured by the logical definition $Do(a|b, s, s') \doteq Do(a, s, s') \vee Do(a, s, s')$.

Flexible behaviours. Our monitoring system is based on a library of Temporal Concurrent Golog scripts representing a set of flexible behaviour fragments. Each of them is associated to a task and can be selected if it is compatible with the execution context. For example a possible behaviour fragment can be written as follows:

$$\begin{array}{l}\mathbf{proc}(explore(d),\\ \ [\pi(t_1, startMap(t_1))\|\pi(t_2, startWand(t_2) :\\ \ \ \pi(t_3, endWand(t_3) : \pi(x, startGoto(x, t_3)) : (t_3 - t_2{<}d)?))].\end{array}$$

This Golog script is associated with the exploration task, it starts both mapping and wandering activities; the wandering phase has a timeout d, after this the rover has to go somewhere. The timeout d will be provided by the calling process that can be either another Golog procedure or a decision of the operator.

Reactive Planner/Interpreter As illustrated before, for each execution cycle, once the status is updated (sensing phase), the Golog interpreter (planning phase) is called to extend the current control sequence up to the planning horizon. When some task ends or fails, new tasks are selected from the task library and compiled into flexible temporal plans filling the timelines.

Under nominal control, the robot's activities are scheduled according to a closed-loop similar to the LOVR (*Localize, Observe general surroundings, look specially for Victims, Report*) sequence in (Murphy 2004). Some of these activities can require the operator initiative that is always allowed.

Failure detection and management Any system malfunctioning or bad behaviour can be detected by the reactive planner (i.e. the Golog interpreter) when world inconsistencies have to be handled. In this case, after an idle cycle a recovery task has to be selected and compiled w.r.t the new execution status. For each component we have classified a set of relevant failures and appropriate flexible (high-level) recovery behaviours. For example, in the visual model, if the scanning processes fails because of a timeout, in the recovery task the pan-tilt unit must be reset taking into account the constraints imposed by the current system status. This can be defined by a very abstract Golog procedure, e.g.

$$\begin{array}{l}\mathbf{proc}(planToPtuInit,\\ \quad \pi(t, time(t)? : plan(2) : \pi(t_1, PtIdle(0) :\\ \qquad time(t_1)? : (t_1 - t < 3)?))).\end{array}$$

In this case, the Golog interpreter is to find a way to compile this procedure getting the pan-tilt idle in less than two steps and three seconds. The planner/Golog interpreter can fail in its plan generation task raising a *planner timeout*. Since the reactive planner is the engine of our control architecture, this failure is critical. We identified three classes of recoveries depending on the priority level of the execution. If the priority is high, a safe mode has to be immediately reached by means of fast reactive procedures (e.g. $goToStandBy$). In medium priority, some extra time for planning can be obtained by interleaving planning and execution: a greedy action is executed so that the interpreter can use the next timeslot to end its work. In the case of low priority, the failure is handled by replanning: a new task is selected and compiled. In medium and low level priority the operator can be explicitly involved in the decision process in a synchronous way. During a high-priority recovery (i.e. $goToStandBy$) the autonomous control is to manage the emergency, unless the operator wants to take care of it disabling the monitoring system.

Mixed-Initiative Planning

The control architecture introduced before allows us to define some hybrid operative modalities lying between autonomous and teleoperated modes and presenting some capabilities that are crucial in a collaborative planning setting. In particular, following (Allen & Ferguson 2002), our system permits *incremental planning*, *plan stability*, and it is also *open to innovation*.

The high-level agent programming paradigm, associated with the short-range planning/interpretation activity, permits an *incremental* generation of plans. In this way, the user attention can be focused on small parts of the problem and the operator can assess local possible decisions, without losing the overall problem constraints.

Plan stability is guaranteed by flexible behaviours and plan recovery procedures, which can harmonize the modification of plans, due to the operator's interventions or exogenous events. Minimal changes to plans lead to short replanning phases minimizing misalignments.

Concerning the *open to innovation* issue, the model-based monitoring activity allows one to build novel plans, under human direction, and to validate and reason about them.

Depending on the operator-system interaction these features are emphasized or obscured. We distinguish among three different mixed-initiative operational modalities.

- **Planning-based interaction.** In this setting, the planning system generates cyclic LOVR sequences and the operator follows this sequence with few modifications, e.g. extending or reducing process durations. Here task dispatching is handled in an automated way and the operator can supervise the decisions consistency minimizing the interventions. The human-operator can also act as an executor and manually control some functional activities scheduled by the planner. For example, he can decide to suspend automated navigations tools and take the control of mobile activities, in this way he can decide to explore an interesting location or escape from difficult environments. In this kind of interaction the operator initiative minimally interferes with the planning activity and *plan stability* is emphasized.
- **Cooperation-based interaction.** In this modality, the operator modifies the control sequence produced by the planner by skipping some tasks or inserting new actions. The operator's interventions can determine a misalignment between the monitoring system expectations (i.e. the control plan) and the state of the system; this is captured at beginning of the next execution cycle when the state monitor provides the current state of the modules. In order to recover the monitor-system adherence, the planner has to start some recovery operations which are presented to the operator. Obviously, these activities are to be executed in real-time by verifying the satisfiability of the underlaying temporal and causal constraints.

  This modality enables maximal flexibility for the planner's and operator's initiatives. Indeed, they can dialogue and work in a concurrent way contributing to the mission completion (*incremental planning*): while the operator tries to modify the plan in order to make it more effective (i.e. the system is *open to innovation*), the monitoring system can validate the operator's choices. Moreover, in the case of safety constraints violations, it warns the user and/or suggests suitable corrections.
- **Operator-based interaction.** This modality is similar to teleoperation, the system activities are directly managed by the operator (some minor autonomy can always be deployed when the operator attention is to be focused on some particular task, e.g. looking for victims). The operator-based interaction is reached when the operators' interventions are very frequent, hence the planner keeps replanning and cannot support the user with a meaningful proactive activity. In this operative scenario, the planner just follows the operators' choices playing in the role of a consistency checker. The monitoring system can notify the user only about safety problems and, in this case, recovery procedures can be suggested (*incremental planning* can be used only to generate non-critical planning procedures).

Each of these modalities is implicitly determined by the way the operator interacts with the system. Indeed, in a mixed-initiative setting, if the operator is idle, the monitor works in the planner-based mode. Instead, the operator's interventions can disturb such a status bringing the system toward the operator-based interaction. However, the operator can always directly set the latter interaction mode by setting to zero the planning horizon and disabling the planner proactive activity. Note that for each mixed-initiative mode, the monitoring system continuously checks the activities performed, including human-operator actions, and when necessary it replans or provides suggestions to the operator.

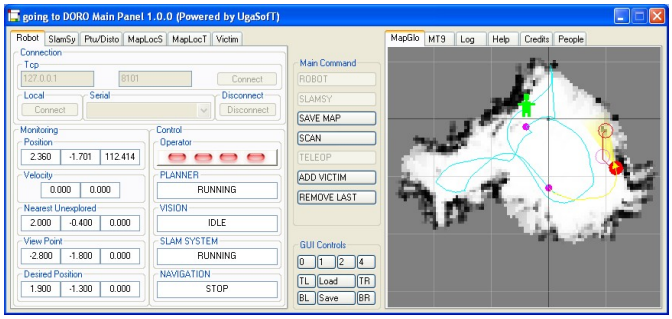

Figure 4: DORO graphical interface showing the current global map, the victims detected and localized, the path-history: in blue the whole history, and in yellow the most recent one.

Mixed-initiative approach at work

The architecture discussed in this article is implemented on our robotic platform (DORO) and here we present some tests performed in a yellow rescue arenas.

Robotic Platfrom. The hardware platform for DORO is a two wheeled differential drive Pioneer from ActivMedia with an on-board laptop hosts navigation, map building, reactive planning routines and the on-board sensors control processing. An additional PC, for remote control, is also used for image processing. The two PCs running Windows XP are linked with an Ethernet wireless LAN (802.11a) to enable remote control and monitoring of the mobile robot. Two color cameras are mounted on top of the robot on a pant-tilt head. A laser range finder DISTO pro is mounted on the pan-tilt between the two cameras.

Robot Software. The robot motion control (speed and heading) and sonar readings are provided by a serial connection to the Pioneer controller using the Aria API facilities. Video streaming and single frames are acquired through the Image Acquisition Toolbox from Matlab (TM). Inertial data and laser measurements are acquired through dedicated C++ modules that manage the low level serial connections.

Experiences in our domestic arenas. We tested the control architecture and the effectiveness of the mixed-initiative approach in our domestic arenas comparing three possible settings: (i) *fully teleoperated*: navigation, slam, and vision

disabled; (ii) *mixed-initiative control*: the monitoring system was enabled and the operator could supervise the rover status and take the control whenever this was needed; (iii) *autonomous control*.

During mixed-initiative control tests, we considered also the percentage of time spent by the operator in *operator-based* mode (see *operator* in the table below). We deployed these three settings on yellow arenas considering increasing surface areas, namely, 20 m^2, 30 m^2, 40 m^2 (see *surface* in the table below), associated with increasingly complex topologies. For each test, there were 4 victims to be discovered. We limited the exploration time to 10 minutes. We performed 10 tests for each modality. Different operators were involved in the experiments in order to avoid an operator visiting the same arena configuration twice.

For each test class we considered: (i) the percentage of the exlored arena surface; (ii) the number of visited and inspected topological environments (rooms, corridors, etc.) w.r.t. the total number; (iii) the overall number of encountered obstacles (i.e. arena bumps); (iv) the number of detected victims; (v) the operator activity (percentage w.r.t. the mission duration). The results are summarized in the Table 1 reporting the average values of each field.

	Fully Teleop			**Supervised**			**Autonomous**		
Surface (m^2)	20	30	40	20	30	40	20	30	40
Explored (%)	85	78	82	85	82	79	49	80	75
Visited env.	5/6	7/9	7/9	6/6	8/9	7/9	3/6	7/9	6/9
Bumps (tot.)	11	7	9	3	2	2	2	1	2
Victims (x/4)	3.0	2.1	2.2	2.5	2.6	2.1	1.3	1.4	1.2
Operator (%)	100	100	100	10	15	15	0	0	0

Table 1: Experimental results for the three operational modalities.

Following the analysis schema in (Scholtz *et al.* 2004) here we discuss the following points: *global navigation*, *local navigation and obstacle encountered*, *vehicle state*, *victim identification*.

Concerning *global navigation*, the performance of the mixed-initiative setting are quite stable while the autonomous system performs poorly in small arenas because narrow environments challenge the navigation system which is to find how to escape from them. In greater and more complex arenas the functional navigation processes (path planner, nearest unexplored point system, etc.) start to be effective while the fully teleoperated behaviour degrades: the operator gets disoriented and often happens that already visited locations and victims are considered as new ones, while we never experienced this in the mixed-initiative and autonomous modes. The effectiveness of the control system for *local navigation* and *vehicle state* awareness can be read on the *bumps* row; indeed the bumps are significantly reduced enabling the monitoring system. In particular, we experienced the recovery procedures effectiveness in warning the operator about the vehicle attitude. E.g. a typical source of bumping in teleoperation is the following: the visual scanning process is interrupted (timeout) and the operator decides to go on in one direction forgetting the pan-tilt in a non-idle position. Enabling the monitor, a recovery procedure interacts with the operator suggesting to reset the pan-tilt position. The victim identification effectiveness can be assessed considering the founded victims in the autonomous mode; considering that visual processing was deployed without any supervision, these results seem quite good (we experienced some rare false-positive).

Our experimental results show that the system performances are enhanced with the presence of an operator supervising the mission. It seems that the autonomous activities are safely performed, but the operator can choose more effective solutions in critical situations. For instance, the number of visited environments in supervised mode (see Table 1) is greater than that one in the autonomous mode, while the victims detected are approximately the same. Furthermore, the number of bumps in teleoperated mode is greater than in both supervised and autonomous settings, and this can be explained by the cognitive workload on the operator during the teleoperation. Thus, we can trade off high performances and low risks by exploiting both human supervision and machine control .

Conclusion

Human-robot interaction and situation awareness are crucial issues in a rescue environment. In this context a suitable interplay between supervised autonomy and full autonomy is needed. For this purpose, we designed a control system where the HRI is fully based on a mixed-initiative planning activity which is to continuously coordinate, integrate, and monitor the operator interventions and decisions with respect to the concurrent functional activities. Our approach integrates model-based executive control, flexible interval planning and high level agent programming.

This control architecture allows us to define some hybrid operative modalities lying between *teleoperated mode* and *autonomous mode* and presenting some capabilities that are crucial in a collaborative planning setting.

We implemented our architecture on our robotic platform (DORO) and tested it in a NIST yellow arena. The comparison between three possible settings (*fully teleoperated*, *mixed-initiative control*, *autonomous control*) produce encouranging results.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Chase, A.; Hsu, J.-J.; Jonsson, A.; Kanefsky, B.; Morris, P.; Rajan, K.; Yglesias, J.; Chafin, B.; Dias, W.; and Maldague, P. 2004. Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission. *Intelligent Systems, IEEE* 19(1):8– 12.

Allen, J., and Ferguson, G. 2002. Human-machine collaborative planning. In *Proceedings of the 3rd international NASA Workshop on Planning and Scheduling for Space*.

Bruemmer, D. J.; Boring, R. L.; Few, D. A.; Marble, J. L.; and Walton, M. C. 2003. "i call shotgun!": An evaluation of mixed-initiative control for novice users of a search and rescue robot. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*.

Burke, J.; Murphy, R.; Coovert, M.; ; and Riddle, D. 2004. Moonlight in miami: A field study of human-robot interaction in the context of an urban search and rescue disaster response training exercise. *Special Issue of Human-Computer Interaction* 19(1,2):21–38.

Burstein, M., and McDermott, D. 1996. Issues in the development of human-computer mixed-initiative planning. *Cognitive Technology* 285–303. Elsevier.

Dautenhahn, K., and Werry, I. 2000. Issues of robot-human interaction dynamics in the rehabilitation of children with autism.

Drury, J. L.; Scholtz, J.; and Yanco, H. A. 2003. Awareness in human-robot interaction. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*.

Finzi, A., and Pirri, F. 2004. Flexible interval planning in concurrent temporal golog. In *Working notes of the 4th International Cognitive Robotics Workshop*.

Finzi, A.; Ingrand, F.; and Muscettola, N. 2004. Model-based executive control through reactive planning for autonomous rovers. In *Proceedings IROS-2004*, 879–884.

Jacoff, A.; Messina, E.; and Evans, J. 2001. A reference test course for urban search and rescue robots. In *FLAIRS Conference 2001*, 499–503.

Kiesler, S., and Hinds, P. 2004. Introduction to the special issue on human-robot interaction. *Special Issue of Human-Computer Interaction* 19(1,2):1–8.

Lang, S.; Kleinehagenbrock, M.; Hohenner, S.; Fritsch, J.; Fink, G. A.; and Sagerer, G. 2003. Providing the basis for human-robot-interaction: a multi-modal attention system for a mobile robot. In *Proceedings of the 5th international conference on Multimodal interfaces*, 28–35. ACM Press.

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

McCarthy, J. 1963. Situations, actions and causal laws. Technical report, Stanford University. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417.

Michael Baker, Robert Casey, B. K., and Yanco, H. A. 2004. Improved interfaces for human-robot interaction in urban search and rescue. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*. "To appear".

Murphy, R. 2004. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man and Cybernetics, Part C* 34(2):138–153.

Muscettola, N.; Nayak, P. P.; Pell, B.; and Williams, B. C. 1998. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence* 103(1-2):5–47.

Muscettola, N.; Dorais, G. A.; Fry, C.; Levinson, R.; and Plaunt, C. 2002. Idea: Planning at the core of autonomous reactive agents. In *Proc. of NASA Workshop on Planning and Scheduling for Space*.

Myers, K. L.; Jarvis, P. A.; Tyson, W. M.; and Wolverton, M. J. 2003. A mixed-initiative framework for robust plan sketching. In *Proceedings of the 2003 International Conference on Automated Planning and Scheduling*.

Pinto, J., and Reiter, R. 1995. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence* 14(2-4):251–268.

Pirri, F., and Reiter, R. 2000. Planning with natural actions in the situation calculus. *Logic-based artificial intelligence* 213–231.

Reiter, R. 1996. Natural actions, concurrency and continuous time in the situation calculus. In *Proceedings of KR'96*, 2–13.

Reiter, R. 2001. *Knowledge in action : logical foundations for specifying and implementing dynamical systems*. MIT Press.

Scholtz, J.; Young, J.; Drury, J.; and Yanco, H. 2004. Evaluation of human-robot interaction awareness in search and rescue. In *Proceedings of the 2004 International Conference on Robotics and Automation*.

Sidner, C., and Dzikovska, M. 2002. Human-robot interaction: Engagement between humans and robots for hosting activities. In *The Fourth IEEE International Conference on Multi-modal Interfaces*, 123–128.

Tadokoro, S.; Kitano, H.; Takahashi, T.; Noda, I.; Matsubara, H.; Shinjoh, A.; Koto, T.; Takeuchi, I.; Takahashi, H.; Matsuno, F.; Hatayama, M.; Nobe, J.; and Shimada, S. 2000. The robocup-rescue project: A robotic approach to the disaster mitigation problem. In *ICRA-2000*, 4089–95.

Tadokoro, S. 2000. Robocuprescue robot league. In *RoboCup-2002*, 482–484.

Volpe, R.; Nesnas, I.; Estlin, T.; Mutz, D.; Petras, R.; and Das, H. 2001. The claraty architecture for robotic autonomy. In *IEEE 2001 Aerospace Conference*.

Williams, B.; Ingham, M.; Chung, S.; Elliott, P.; Hofbaur, M.; and Sullivan, G. 2003. Model-based programming of fault-aware systems. *AI Magazine*.

Yanco, H., and Drury, J. 2002. A taxonomy for human-robot interaction. In *Proc. AAAI Fall Symposium on Human-Robot Interaction*, 111–119.