

# Human-robot interaction through mixed-initiative planning for rescue and search rovers

AndreA Orlandini


*Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*

## Cite this paper

Downloaded from [Academia.edu](#) 

[Get the citation in MLA, APA, or Chicago styles](#)

## Related papers

[Download a PDF Pack](#) of the best related papers 



[Handling Continuous-Valued Attributes in Incremental First-Order Rules Learning](#)

AndreA Orlandini

[A mixed-initiative approach to human-robot interaction in rescue scenarios](#)

Alberto Finzi

[Model-based rescue robot control with ECLiPSe framework](#)

AndreA Orlandini

# Human-Robot Interaction through Mixed-Initiative Planning for Rescue and Search Rovers

[Alberto Finzi](#)<sup>1</sup> and [Andrea Orlandini](#)<sup>2</sup>

<sup>1</sup> Dipartimento di Informatica e Sistemistica  
Università degli Studi di Roma “La Sapienza”

<sup>2</sup> Dipartimento di Informatica e Automazione  
Università degli Studi di Roma TRE

**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 interventions and the concurrent activities of a rescue rover. 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 in rescue missions. We implemented this control architecture on a robotic system (DORO) and tested it in rescue arenas comparing its performances in different settings.

## 1 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, NIST (together with the Japan National Special Project for Earthquake Disaster Mitigation in Urban Areas) [24, 23, 11, 8] 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 environment, discussed in [4] and [14] concerns operator situation awareness and human-robot interaction (HRI). In [14] the difficulties in forming a mental model of the “robot eye” are endorsed, with marking into evidence the role of the team. Differently from real tests, like the one in Miami (see [4]), during rescue competitions the operator is forced to be alone while coordinating the robot activities, as 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: following 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 [3]: the operator attention has to be focalised over a wide range of activities, loosing concentration on the real rescue mission objective: find 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 [9, 27, 7, 13, 27].

In this work we describe a mixed-initiative planning approach [1, 16, 2, 5] to HRI in a rescue domain and illustrate the main functionalities of a rescue robot system<sup>3</sup>. 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 continuous reactive planning process which is to: i. check the execution status with respect to a declarative model of the plan; 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 [7]:

- robot-human: given a declarative model of the robot activities, the monitoring system can be “self-aware” about the current situation, at different levels of abstraction; in this way, complex and not nominal interactions among activities can be detected and displayed to the operator;
- human-robot: 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 [1].

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

## 2 Rescue Scenario

The National Institute of Standard Technology (NIST) has developed physical test scenarios for rescue competitions. There are three NIST arenas, denoted by yellow, orange, and red of varying degree of difficulty. Yellow arena represents an indoor flat environment with minor structural damage (e.g. overturned furniture), the orange arena is multilevel and have more rubble (e.g. bricks), the red one represents a very damaged environment, unstructured: 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 competition is a very hard test bed for robots and their architectures. In fact, the operator-robot has to coordinate several activities: explore and map the environment, avoiding obstacles (bumping is severely penalized), lo-

---

<sup>3</sup> Doro is the third award winner in Lisbon contest (2004)

calize itself, search for victims, correctly locate them on the map, identify them through a numbered tag, and finally describe her 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. This 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, [14] propose a high level sequence tasks cycle as reference for the rescue system behaviour such as Localize, Observe general surroundings, look specially for Victims, Report (LOVR). Our cycle's interpretation corresponds to the following tasks sequence: map construction, visual observation, vision process execution and victim's presence report.

### 3 Human Robot Interaction and Mixed Initiative Planning

There are 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. [6, 9, 4, 22, 10] and the already cited [14, 13, 27, 7], 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 autonomous drawn map, and an 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. This 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:

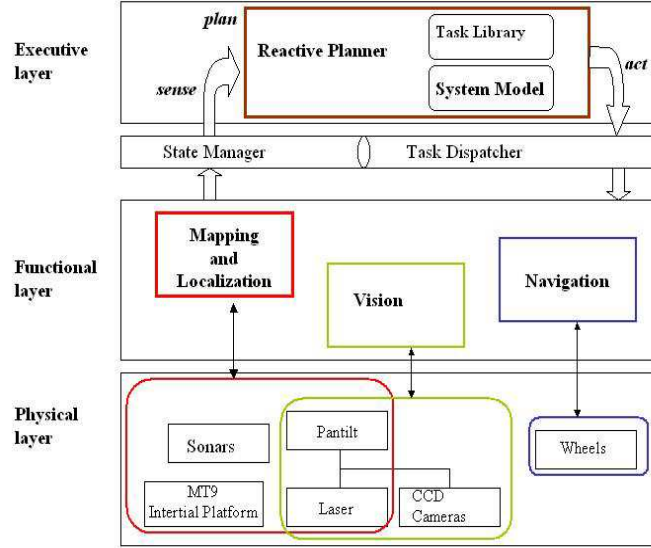
- **Model-based control.** The control system is endowed with declarative models of the controllable activities where causal and temporal relations are explicitly represented [15, 26]. In this way, hard and soft constraints can be directly encoded and monitored. Furthermore, formal methods and reasoning 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 activities are explicitly modeled and supervised by the control system; on the other hand, the model-based monitoring activity exports a view of the system 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 interventions by continuously performing sense-plan-act cycles. At each cycle the reactive planner is 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, also 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, the time and resources needed cannot be rigidly scheduled, instead it is necessary to account for flexible behaviours, allowing 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 bind, instead this value spans a temporal interval.
- **High-level agent programming.** The high-level agent programming paradigm allows 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 the control knowledge is encoded by abstract and partial procedures. Both system's and 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.

## 4 Control Architecture

In this section we describe the control system we have defined to incorporate the design principles introduced above. Following the approach in [15, 26, 25] 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 1 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 manages the communication between the executive and the functional layer.

The *state manager* gets from each single module its current status so that any module can query the state manager about the status of any another module. The state manager updates its information every 200 msec., the task dispatcher is to send 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 is to produce a plan of actions (planning phase about 0.5 sec.) and can yield the first set of commands to the *task dispatcher*. In the execution phase (about 0.5 sec.), each module can read the signals and start 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 canceled and a suitable recovery procedure is provided.



**Fig. 1.** Control architecture

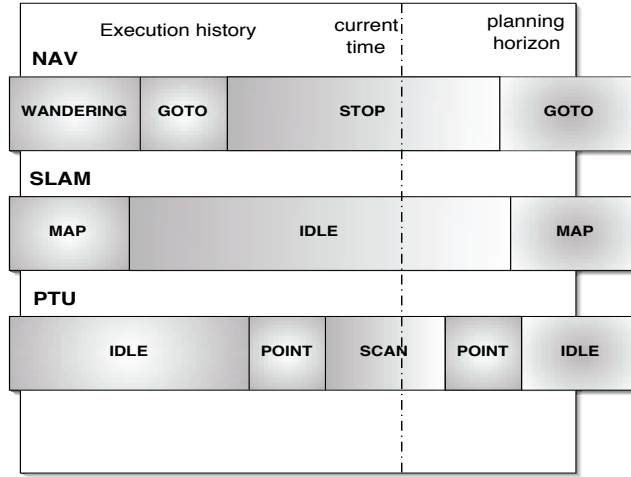
## 5 Model-Based Monitoring

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 is to harmonize 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. We implement these features by deploying *high-level agent programming* in Temporal Concurrent Golog [20] which provides both a declarative language (i.e. Temporal Concurrent Situation Calculus [17, 19, 18]) to represent the system properties and the planning engine to generate the control sequences.

*Temporal Concurrent Situation Calculus.* The Situation Calculus (*SC*) [12] 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 [17, 19, 18] 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 DORO processes and states are explicitly represented by a declarative dynamic-temporal model specified in the Temporal Concurrent

Situation Calculus (TCSC) . This model represents the cause-effects relations and the 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 our rescue system the components are: *pan-tilt*, *slam*, *navigation*, *visualPerception*. 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*. E.g. Figure 2 illustrates the evolution of *navigation*, *slam*, and *pan-tilt* up to a planning horizon.



**Fig. 2.** Timelines evolution

Hard time constraints among the 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 to denote procedural scripts composed of the primitive actions explicitly represented in a SC action theory. This hybrid framework integrates both 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:

```

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

```

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 nondeterministic choices to the interpreter. More in general, a Golog script can range from a completely defined procedural program to an abstract general purpose planning algorithm like the following:

```

proc(plan( $n$ ), true? |  $\pi(a, (primitive\_action(a))? : a) : plan(n - 1)$ )

```

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

*Flexible behaviours.* Our monitoring system is based on a library of Temporal Concurrent Golog scripts representing a set of flexible behaviour fragments. Each of these 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:

```

proc(explore( $d$ ),
  [ $\pi(t_1, startMap(t_1)) || \pi(t_2, startWand(t_2) :$ 
     $\pi(t_3, endWand(t_3) : \pi(x, startGoto(x, t_3)) : (t_3 - t_2 < d)?)$ ]).

```

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. This timeout  $d$  will be provided by the calling process that can be either another Golog procedure or an operator decision.

*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 [14]. 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.

$$\text{proc}(\text{planToPtUnitInit}, \pi(t, \text{time}(t)? : \text{plan}(2) : \\ \pi(t_1, \text{PtIdle}(0) : \text{time}(t_1)? : (t_1 - t < 3)?))).$$

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 itself its plan generation task and we have 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 time-slot 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*) we have no mixed initiative, if the operator wants to take care of it the monitoring system is bypassed.

## 6 Mixed-Initiative Planning

The control architecture introduced before allows us to define some hybrid operative modalities lying between autonomous mode and teleoperated and presenting some capabilities that are crucial in a collaborative planning setting. In particular, following [2], 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 consider possible options on them, without losing the overall problem constraints. *Plan stability* is guaranteed by flexible behaviours and plan recovery procedures, which can harmonize plan's modification, due to the operator interventions or exogenous events. Minimal changes of plans leads to short replanning phases minimizing misalignments. Concerning the *open to innovation* issue, the model-based monitoring activity allows 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 a LOVR sequences and the operator follows this sequence providing little modifications, e.g. extending or reducing the processes duration. 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 underlying temporal and causal constraints. This modality enables the maximal flexibility for the planner and operator interactive 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, and, in the case of safety constraints violations, warning the user and/or suggesting 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 and 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 not 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 this status bringing the system toward the operator-based interaction. However, the operator can always directly set the latter interaction mode by annulling the planner proactive activity (and the planning horizon). 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.

## 7 Mixed-initiative approach at work

We implemented our architecture on our robotic platform (DORO) and tested it in the Rescue Arenas (yellow type).

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 a 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 pan-tilt head. A laser range finder DISTO pro, is mounted on the pan-tilt between the two cameras.

The robot motion control (speed and heading) and sonar lectures 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.

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 the 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 three kind of 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 of these modalities. For each test class we considered: i. the percentage of the arena surface explored; ii. the number of topological environments (rooms, corridors, etc.) visited and inspected w.r.t. the total number; iii. the overall number of obstacles encountered (bumps); iv. the number of victims found; v. the operator activity (percentage w.r.t. the mission duration). The results are summarized in the following table.

	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

Following the analysis schema in [21] 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 one, instead, 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 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 victims found in the autonomous mode, considering that visual processing was deployed without any supervision, these results seem quite good (we experienced some rare false-positive).

## 8 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*) produced encouraging experimental results.

## References

1. M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.C.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, Kanna Rajan, J. Yglesias, B.G. Chafin, W.C. Dias, and P.F. Maldague. Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission. *Intelligent Systems, IEEE*, 19(1):8–12, 2004.
2. James Allen and George Ferguson. Human-machine collaborative planning. In *Proceedings of the 3rd international NASA Workshop on Planning and Scheduling for Space*, 2002.
3. David J. Bruemmer, Ronald L. Boring, Douglas A. Few, Julie L. Marble, and Miles C. Walton. "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*, 2003.
4. J. Burke, R.R. Murphy, M. Covert, , and D. Riddle. 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, 2004.
5. Mark Burstein and Drew McDermott. Issues in the development of human-computer mixed-initiative planning. *Cognitive Technology*, pages 285–303, 1996. Elsevier.
6. K. Dautenhahn and I. Werry. Issues of robot-human interaction dynamics in the rehabilitation of children with autism, 2000.
7. J. L. Drury, J. Scholtz, and H. A. Yanco. Awareness in human-robot interaction. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, October 2003.
8. Adam Jacoff, Elena Messina, and John Evans. A reference test course for urban search and rescue robots. In *FLAIRS Conference 2001*, pages 499–503, 2001.

9. Sara Kiesler and Pamela Hinds. Introduction to the special issue on human-robot interaction. *Special Issue of Human-Computer Interaction*, 19(1,2):1–8, 2004.
10. Sebastian Lang, Marcus Kleinhagenbrock, Sascha Hohener, Jannik Fritsch, Gernot A. Fink, and Gerhard Sagerer. 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*, pages 28–35. ACM Press, 2003.
11. Bruce A. Maxwell, William D. Smart, Adam Jacoff, Jennifer Casper, Brian Weiss, Jean Scholtz, Holly A. Yanco, Mark Micire, Ashley W. Stroupe, Daniel P. Stormont, and Tom Lauwers. 2003 aaai robot competition and exhibition. *AI Magazine*, 25(2):68–80, 2004.
12. J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963. Reprinted in *Semantic Information Processing* (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410–417.
13. Brenden Keyes Michael Baker, Robert Casey and Holly A. Yanco. Improved interfaces for human-robot interaction in urban search and rescue. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 2004. "To appear".
14. R.R. Murphy. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 34(2):138–153, 2004.
15. N. Muscettola, G. A. Dorais, C. Fry, R. Levinson, and C. Plaunt. Idea: Planning at the core of autonomous reactive agents. In *Proc. of NASA Workshop on Planning and Scheduling for Space*, 2002.
16. Karen L. Myers, Peter A. Jarvis, W. Mabry Tyson, and Michael J. Wolverton. A mixed-initiative framework for robust plan sketching. In *Proceedings of the 2003 International Conference on Automated Planning and Scheduling*, 2003.
17. J.A. Pinto and R. Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):251–268, September 1995.
18. Fiora Pirri and Raymond Reiter. Planning with natural actions in the situation calculus. *Logic-based artificial intelligence*, pages 213–231, 2000.
19. R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proceedings of KR'96*, pages 2–13, 1996.
20. Raymond Reiter. *Knowledge in action : logical foundations for specifying and implementing dynamical systems*. MIT Press, 2001.
21. J.J. Scholtz, J. Young, J.L. Drury, and H.A. Yanco. Evaluation of human-robot interaction awareness in search and rescue. In *Proceedings of the 2004 International Conference on Robotics and Automation*, April 2004.
22. C. Sidner and M. Dzikovska. Human-robot interaction: Engagement between humans and robots for hosting activities. In *The Fourth IEEE International Conference on Multi-modal Interfaces*, pages 123–128, 2002.
23. Satoshi Tadokoro. Robocuprescue robot league. In *RoboCup-2002*, pages 482–484, 2000.
24. Satoshi Tadokoro, Hiroaki Kitano, Tomoichi Takahashi, Itsuki Noda, Hitoshi Matsubara, Atsushi Shinjoh, Tetsuhiko Koto, Ikuo Takeuchi, Hironao Takahashi, Fumitoshi Matsuno, Michinori Hatayama, Jun Nobe, and Susumu Shimada. The robocup-rescue project: A robotic approach to the disaster mitigation problem. In *ICRA-2000*, pages 4089–95, 2000.
25. R. Volpe, I.A.D. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das. The claraty architecture for robotic autonomy. In *IEEE 2001 Aerospace Conference*, March 2001.
26. B. Williams, M. Ingham, S. Chung, P. Elliott, M. Hofbaur, and G. Sullivan. Model-based programming of fault-aware systems. *AI Magazine*, Winter 2003.
27. H. Yanco and J. Drury. A taxonomy for human-robot interaction. In *Proc. AAAI Fall Symposium on Human-Robot Interaction*, pages 111–119, 2002.