Situation aware rescue robots

Conference Paper · July 2005

DOI: 10.1109/SSRR.2005.1501270 · Source: IEEE Xplore

Situation Aware Rescue Robots

A. Carbone, A. Finzi, A. Orlandini, F.Pirri, G. Ugazio,
DIS, University of Roma La Sapienza-DIA University of Roma-3

***Abstract*— We present a model-based approach to the execution and control of an autonomous system, based on flexible behaviours, and supporting a novel view of human-robot interaction. The well-known RoboCup Rescue competition is also discussed and presented as a special case for testing human-robot interaction architectures. The activities of the system, according to the components performing them, are modeled as flexible behaviours and are executed according to different modalities, such as fully operated, supervised, fully autonomous. We finally discuss the implementation and tests done both during the contests and in the laboratory, to show performances according to the different modalities.**

I. 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 ([1], [2], [3], [4]), 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 [5] and [6] concerns the operator situation awareness and human-robot interaction. In [6] 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 [5]), 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: 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 [7]: 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 [8], [9], [10], [11], [9].

Fig. 1. The mobile robot DORO, in a yellow arena built in the ALCOR Laboratory.

In this work we describe a model-based approach to flexible behaviours (i.e. behaviours spanning over flexible elapse of time) and describe the main functionalities of a rescue robot system[1], together with their interleaving and concurrent processes. We show how this model supports human-robot interaction for the execution and control of the diverse activities needed during a complex competition such as the rescue one. The advantage and novelty of this approach can be appreciated considering the HRI awareness discussed in [10]:

- robot-human interaction: the system is "self-aware" about the current situation, at different levels of abstraction, 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 her attention toward the current state of exploration, while she interacts with a mixed initiative reactive planner [12].

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.

II. The Rescue Scenario

NIST has developed physical test scenarios for RoboCup Rescue competitions. There are three NIST arenas, denoted

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

by yellow, orange, and red of varying degrees 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: multi-level, 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. Victims are dummies or thermal/audio identifications. In order to play the game the operator-robot has to coordinate several activities: explore and map the environment, avoiding obstacles (bumping is severely penalized) localize 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 game session 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.

III. Control Architecture

Several activities need to be coordinated and controlled during a mission, and the interface is one of them. A model of execution is thus a formal framework allowing for a consistent description of the correct timing of any kind of behaviour the system has to perform to successfully conclude a mission. However as the domain is uncertain, the result of any action can be unexpected, and the time and resources needed cannot be rigidly scheduled, it is necessary to account for flexible behaviours, which means managing dynamic change of time and resource allocation. In this section we describe the model underlying the flexible behaviours approach, mentioning the coordination and control of processes that are described in more details in the next sections, to give implementation concreteness to our formal framework.

A model-based executive control system [13], [14] supervises and integrates both the robot modules activities and the operator interventions. Following this approach, the main robot and operator processes (e.g. mapping, laser scanning, navigation etc.) are explicitly represented by a declarative temporal model (see Section IV) which permits a global interpretation of the execution context. Given this model, a reactive planner can monitor the system status and generate the control on the fly continuously performing sense-plan-act cycles. At each cycle the reactive planner is to: generate the robot activities up to a planning horizon, and monitor the consistency of the running activities (w.r.t. the declarative model) managing failures. The short-range planning activity can balance reactivity and goal-oriented behaviour: short-term goals/tasks and external/internal events can be combined while the reactive planner tries to solve conflicts. In this setting, also the human operator can interact with the control system influencing the planning activity in a mixed initiative manner (analogously to [12]). Figure 2 illustrates the overall control architecture designed for DORO.

The physical layer, composed by the robot and all the effectors and sensors, is controlled by a set of functional modules. These modules devise to the *decision daemons* informations that are processed in order to perform different tasks, e.g. construct the map or calculate an exploration path. The *state manager* and *task dispatcher* in the figure are designed to manages the communication between the executive and the other layers.

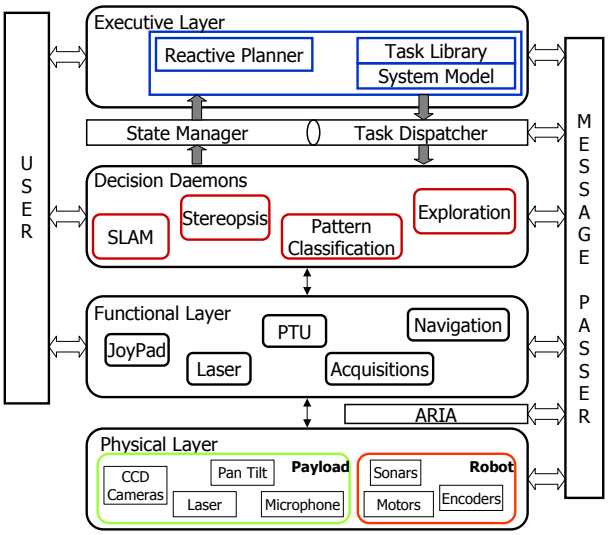

Fig. 2. Control architecture

The state manager gets from each single daemon its current status so that it is possible query the state manager about the status of any daemons. The state manager updates its information every 200 msec. The task dispatcher sends tasks activation signals to the daemons (e.g. $start\_map$) upon receiving requests from the planner. The overall computational cycle works as follows: the planner gets the daemons 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 yields the first set of commands to the task dispatcher. In the execution phase (about 0.5 sec.), each daemons reads the signals and starts its task modifying its state and using the functional modules data. When the next cycle starts, the planner reads the updated status through the state manager and checks 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.

IV. Model-Based Monitoring

A model-based monitoring system is to enhance the situation awareness of the whole system. 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. The reactive planner was developed following an high level programming approach, deploying the Temporal Concurrent Golog [15]: temporally flexible plans are produced by a Golog interpreter which completes partially specified behaviours (Golog programs) selected from a plan library.

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.

a) Declarative Model: The main processes and states of DORO are explicitly represented by a declarative dynamic-temporal model specified in the Temporal Concurrent Situation Calculus [15]. The 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 (analogously to Constraints Based Interval Planning paradigm [16]). Each component 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*, *rangeFinder*, *slam*, *navigation*, *visualPerception*, etc. Both states and activities are represented by temporal intervals. Each of these is associated with a set of processes, e.g. *navigation* can be going at speed s or stop: Nav_stop, $Nv\_moving(s)$; *pan-tilt* can either be idling in position $\vec{\theta}$ ($pt\_idle(\vec{\theta})$), moving toward $\vec{\theta}$ ($pt\_moving(\vec{\theta})$), or scanning ($pt\_scanning(\vec{\theta})$). The history of states for a component over a period of time is a *timeline* (see Figure 3).

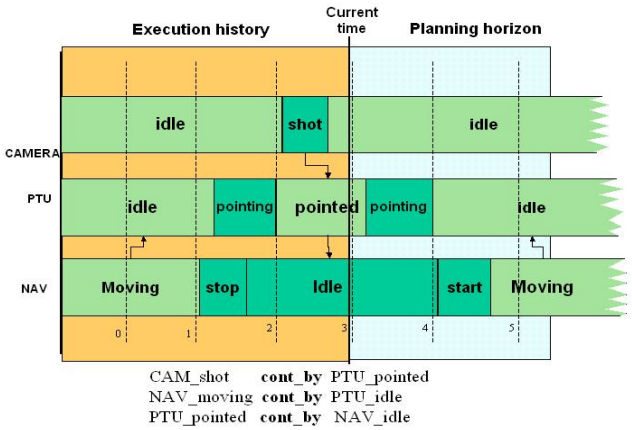

Fig. 3. Timeline evolution

Hard time constraints among the activities can be defined by a temporal model using Allen-like temporal relations, e.g.: $Pt\_moving(\vec{\theta})$ *precedes* $Pt\_scanning(\vec{\theta})$, $Pt\_scanning(\vec{\theta})$ *during* Nv_stop, etc..

b) Flexible behaviours: Each component will be associated with a library of flexible Golog scripts representing possible behaviours which are selected by the monitoring system considering the execution context and the goals. For example, given the PTU state variable, a possible behaviour can be written as follows:

$$\begin{aligned}&\mathbf{proc}(plan\_Pt,\ \pi(t,\pi(t',(\exists x.Md\_map(x,t))?|\\&\quad(\exists x.Md\_search(x,t))?\ :\ wait\_location\ :\\&\quad\quad Ptscan\ :\ PtIdle:(time=t'\wedge t-t'\leq d)?)),\end{aligned}$$

where $wait\_location$: $Ptscan$: $PtIdle$ are three Golog procedures defining the expected pant-tilt behaviours during the search mode. The final test enforces a maximal d time duration for the whole procedure execution.

c) Reactive Planner/Interpreter: As illustrated in Section III, 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 [6].

d) Failure detection and management: Any system malfunctioning or bad behaviour can be detected by the reactive planner when world inconsistencies have to be handled. After an idle cycle a recovery task has to be selected and compiled w.r.t the new execution status. Each component is associated with a class 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.

$$\begin{aligned}&\mathbf{proc}(planToPtuInit,\\&\quad\pi(t)[plan(t,2):(\exists t_1.Ptu\_idle(t_1))?\wedge t_1-t<3]).\end{aligned}$$

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, that is, its plan generation task, in case of 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 is 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.

V. Processes and Components

The goal of the rescue mission is victim finding, identification and positioning. Identification means 1. Find the victim and the associated numbered label; 2. possibly identify the victim state and condition. The first task is relatively doable autonomously, as it requires to suitably explore the arena to find a victim either with the camera (the victim is visible) or with sound (the victim can be heard) or infrared (detect heat),

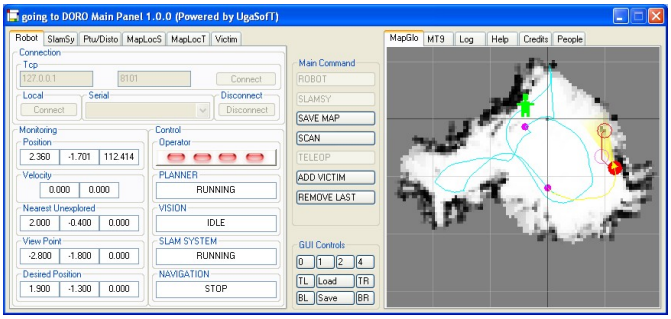

Fig. 4. Interface with the path highlighted, in blue the whole path-history, and in yellow the most recent path. An indication of the nearest unexplored (NU) zone appears on the panel as an arrow pointing towards this direction (see the dark red arrow on the map; the red ribbon indicates the UP).

find the label and identify, possibly with an OCR, the number, and finally project the victim position on the map. As far as the second task is involved, the system should report on whether the victim is on surface, trapped, or entombed, and whether her state is conscious or not, emitting sounds or silent, moving or not, and also the type of mouvement: thrilling, waving etc.. Some of these tasks can be autonomously performed, but most need an accurate robot-operator interaction for coordinating sensor data interpretation. Actually, the second task is not completely achievable autonomously, and definitely needs the operator to look at the victim through the camera and verify which parts are occluded (the head, the legs, the whole body etc..). In this section we shall therefore describe only the first component of victim identification, and the extent to which it can be successfully achieved. Since the exploration has to be achieved in a specified lapse of time, namely 20 minutes, given that the arena is about 40 square meters, with several paths and choices, a good strategy is to solicit the operator attention toward a given direction, leaving to her a more accurate analysis, e.g. about the victim condition. Note that identification is rather involved for light conditions (see Figure 5, taken at the Lisbon rescue); in the last competition (Lisbon 2004) covered areas, were mainly built in the yellow and orange arenas. In this section we detail the components and the processes controlled by the system, to support the operator choices for victim identification.

A. Interactive exploration and nearest unexplored position

We conceive two maps the *Global Map* GM and the *Local Map* (LM), this last is obtained at each SLAM-cycle, via sonar sensor fusion; the LM is incrementally integrated into the GM, exploiting the well known Bayesian approach (see [17]). Integration follows a correct alignment of the robot with the map, suitably reviewing odometric errors. To make the map worth for the operator, so that she can follow the path pursued so far, and verify whether some region has not yet been visited, we introduce two concepts the History Path (HP) and the Exploration Path (EP), the first (see the blue ribbon in Figure 4) illustrates the whole path threaded from the very beginning, and the latter (look at the yellow ribbon) highlights the last steps taken. A typical scenario where the paths are useful is when it is necessary to draw conclusions like “this area has already been visited”, “the victim currently observed is inaccessible from this path, take a snapshot and try to reach it from another way”, and so on.

e) End of Exploration Test: EP is also useful to determine the end of the exploration phase in the LOVR sequence: we define an *End of Exploration Test* as a function of time and the data cumulated from t_0 to t_n (current time), where t_0 is the starting time of the current exploration phase. In this way exploration might stop according to: i. time elapsed since exploration start; ii. area explored during current exploration; iii. exploration path length. Given that $A(t_n)$ is the number of cells (the area) explored at time t_n during the current exploration phase, the amount of work done at time t_n is definable as: $\mathcal{W}(t_n) = \alpha\ |EP(t_n)| + \beta\ A(t_n) + \gamma\ t_n$, where α, β and γ are normalization parameters. When $\mathcal{W}(t_n) > \tau$ (τ is the learning rate) the exploration phase is over, EP is collected in RP (and cleaned) and time is set to t_0.

f) Nearest Unexplored Position: While $W(t) < \tau$ the *nearest unexplored* region is evaluated assigning a score to each cell of the current GM, and then by an optimized version of *Value Iteration Algorithm* [18], [19] the minimum cost path (Unexplored Path or UP) toward the unexplored cell is found. An indication of the nearest unexplored (NU) zone appears on the panel as an arrow pointing towards this direction (see Figure 4). Our optimized version of the Value Iteration Algorithm is based on updating on-line cell values using yet-updated neighbour values. So value propagation gently follows a cell index cycle. To avoid counter-current value propagation (when a cell value needs to be updated using cells that has to be scanned) map cells are cycled from all eight planar wind directions, and only a rectangular box, containing all modified value and their neighbors, is updated each cycle.

g) View Point: When $W(t) > \tau$, the exploration is over and then the search looks for an optimal vantage point, i.e. the spot to reach and the direction of sight. We will call this position View Point (VP). The View Point is obtained maximizing a functional $\mathcal{J}(s)$ defined over the map cell of the last explored area (the map cells that has changed occupancy value in last exploration phase):

$$J(s) = \sum_{i=1}^{4} \lambda_i \phi_i(s), \quad VP = \arg\max_s J(s).$$

The above four components, w.r.t. the VP to be found, are:

1. Distance from the gravity center: the VP distance from the center of gravity of the area explored so far, is to be minimized (motivation: robot shouldn’t forget to analyze the whole explored area).
2. Distances in a view path: the VP distance from each point in the view path (i.e. the set of VP found previously) is to be maximal (motivation: robot shouldn’t lose time to watch where it has yet watched).
3. Distance from the current position: the VP distance from the current robot position is to be minimal (motivation: robot shouldn’t spend so much energy and time to go to the VP).

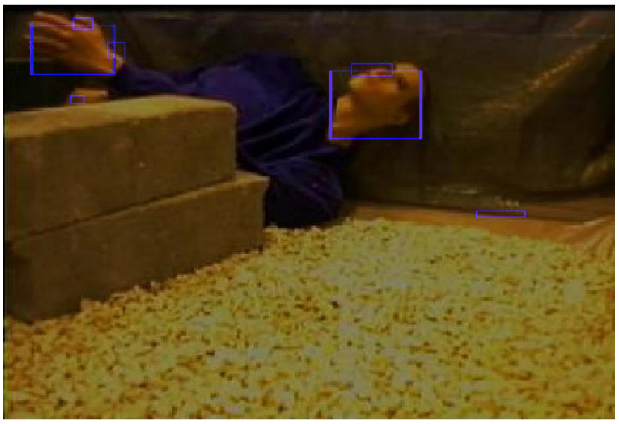

Fig. 5. Victim recognition. Skin textures particles obtained by on-line re-sampling for mixture models, are clusterized and a skin region is highlighted. Picture from the Lisbon red arena

4. Quality: the area to which VP belongs is to be free from obstacles and with a wide visual angle (motivation: robot shouldn't place itself in a rough or hidden area).

The coefficients λ_i can be supposed constant for this simple analysis. One of our goals is to make them adaptive as a function of specific contextual properties, or learned. Maximization is obtained by Hill Climbing, with random restart. Given the VP, the robot primary objective is to reach it. So, while it has not yet reached its goal, the View Point Path (VPP) is obtained using Optimized Value Iteration. The black arrows in Figure 4 illustrate the View Point and the red ribbon indicates the View Point Path. In both cases (View Point and Nearest Unexplored) a partial goal, called Desired Position (DP), is computed (the light red arrow in Figure 4) to drive the robot through the right position.

B. Victim perception and positioning

The activity of the camera (and the microphones) are controlled by the executor and it is launched when a seemingly good position is found, according to the previously described exploration strategy. The result of the *perceptual* process (described in this and the next paragraphs) is published to the PTU-positioning, which – in terms of the processes timing and interaction – means that the Camera and Sound processes must be concurrent and both must meet the PTU-process.

h) Skin classification: In this paragraph we briefly overview the algorithm for skin recognition. We have chosen to segment the image according to the presence of skin features, and to represent skin-features in the RGB color space, augmented with the gradient over the three channels. The recognition process is based upon the construction of a multivariate Gaussian mixture, using the well known EM algorithm (off-line trained while missions are suspended) and on a particular adaptation process that can cope with extreme light conditions. To complete its training activity the robot can visit the arenas before "earthquake", so it can take several shots of the victims. On the basis of the shots, pieces of skin from the pictures are collected all together to form a *skin map*, note that several skin maps can be built during a whole competition. In general, by our experience mixing the maps do not improve recognition performance. The skin map is then used for building a multivariate Gaussian, using the the well known iterative expectation maximization (EM) algorithm [20], based on the implementation in Netlab [21]. Multivariate Gaussian mixture for modeling human skin has been shown to give nice results (e.g. see [22]). Features are modeled by a multivariate Gaussian mixture (GMM): $g_{skin}(\mathbf{x}) = \sum_{k=1}^{M} f(k)p(\mathbf{x}|k)$. Here $\mathbf{x} \in \mathbb{R}^6$ is the 6-dimensional feature, and the $f(k)$, $0 \leq f(k) \leq 1$, $\forall k$, are all *mixing coefficients*. Depending on light conditions the mixture number can substantially vary, therefore two steps have to be taken: 1. Build a skin map for interiors, one for pseudo-exteriors etc. 2. Define an adaptation procedure, so that the same map can be used. Since the two methods are not exclusive (the first is rather boring because is not completely autonomous), we have experienced both, build a skin map as soon as possible, and rely on adaptation. Given the multivariate Gaussian mixture built on the current map, the feature classification process needs to be further separated into two steps, the first step requires an on-line training to model the multivariate mixture generating the current observation $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$, $\mathbf{y} \in \mathbb{R}^6$; here the number of components are decided according to the number of peaks found in the image histogram, according to a suitable threshold. In the second step those parameters are chosen, from the on-line mixture, maximizing the expectation to see some components of the known mixture g_{skin}. The basic idea is that if skin features are observed in the current image, then some of the components in the previously learned model of the skin, must be more likely, given the image. Namely, let I be the GMM modeling the image, with H components, and S the GMM modeling the skin (with M components), then the log-likelihood of the k-th component of S, given I is $\sum_j \log \mathcal{N}(\mu_k - \mu_j, \Sigma_j)$, where μ_j and Σ_j, $j = 1, .., H$, are the mean and covariance of the j-th component of I. Maximizing the log-likelihood (i.e. minimizing the error):

$$\mathcal{L}(\Theta_i, \Theta_j) = \arg \max_{(i,j)} \sum_{i=1}^{M} \sum_{j=1}^{H} \log \mathcal{N}(\mu_i - \mu_j, \Sigma_j)$$

We obtain the new set of parameters Θ_i and Θ'_j for the i-th component of S and the j-th component of I. Taking the components maximizing the log-likelihood up to a given threshold ($max - 2$), the final classification is achieved using Bayes decision rule, i.e. $\mathbf{x} \in Skin$ iff $\pi(Skin|\mathbf{x}, \Theta^c) > \pi(C|\mathbf{x}, \Theta'_j), \forall j = 1, ..., H - c$, here $Skin$ is the class modeled by S, C is the class modeled by I, Θ^c are the parameters chosen for S, and Θ'_j are the parameters of the j-th component of I, and excluding those that have been selected as "similar", to build Θ^c. On-line classification is really needed when the mission starts, and whenever there are drastic changes in lighting condition, to induce adaptation. Therefore to make features classification effective, unless the above mentioned circumstances occur (which are devised comparing the image

histograms), then the parameters for classification are set to those used in the previous step, i.e. for all $\theta \in \Theta$, $\theta_{i,t} = \theta_{i,t-1}$, given that t_0 is the first observation in the arena, after the mission started. The results of this post-conditioned classification, modeled by a multivariate Gaussian mixture, is quite satisfying (see the results on a quite difficult lighting condition as illustrated in Figure 5), but still needs improvements: on image with a resolution of 72dpi and dimension 320×240, it takes 6.607 cpu time. False positives constitute the 20% of the total number of pixels classified, and false negatives, i.e. the number of pixel that should have belonged to $Skin$ but where not classified is around 45% of the total number of image pixels, that would be classified as belonging to the $Skin$ class.

i) Sound Interface: The perceptual framework also incorporates sound source information, which is essential because some victim is entombed or hidden in unreachable corners. The two Sony far-field microphones are processed by the $Matlab$ functions designed for acquiring and recording sound, trying to detect close signals emerging from background noise.

j) Positioning the victims on the map: According to the described processes, as soon as a victim is found, her position must be determined to incorporate it into the GM. The available information concerns: the robot position and orientation, the head position (pan and tilt angle) and range finder measurements, and the robot height. If the recognized victim head is not in the center of the map, a step of "pointing" is required to calculate the desired pan/tilt position.

VI. Supervised Control at work in Rescue Arenas

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

A. DORO platform

The hardware platform for DORO is a two wheeled differential drive Pioneer 3DX from ActivMedia with an on-board low consuming laptop that hosts navigation, map building, reactive planning routines, and the on-board sensors control and for sound 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 pant-tilt head, images are acquired with a frame grabber through a firewire IEEE1394. Two far-field microphones are located at the front of the platform to detect victim's sound. A laser range finder DISTO pro, is mounted on the pan-tilt between the two cameras. An inertial platform MT9 from XSense is positioned at the geometric center of rotation and delivers high precision acceleration and heading values that are used to enhance the dead-reckoning.

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.

B. Experiences in the Rescue Competition

We deployed our architecture (enabling only a subset of the functionalities described above, depending on the context) at the RoboCup Real Rescue 2004 competition. We performed six missions: two preliminary rounds, two semi finals, and two final contests. Following the analysis schema in [23] here we discuss the following points:

k) Global Navigation: The mapping and localization system was a very effective support for exploration. Except for the first round (where we encountered a warm up problem), for all the other missions we could explore and map more than the 80% of the arena's environments (rooms, niches, ails etc.).

l) Local Navigation and obstacle encountered: We had just one minor incident (obstacle encountered) during the first round. In this context, we found the 2DOF pan-tilt unit very effective for navigation in difficult environments (this is observed also in [23]), since it allows a visual inspection of the robot attitude w.r.t. the surrounding environment enhancing the overall operator perception of the vehicle volume.

m) Vehicle State: During the rescue rounds, the planning activity was reduced and used only to monitor the nominal control execution. In the case of misalignments, the operator was warned, but no recovery process was enabled. Even in this minimal setting, the monitoring activity supported the operator awareness of the vehicle executive status. E.g., a simple but frequent problem signaled by the monitoring system was the following: many times the user manually disabled the mapping system while searching for victims, forgetting to restart it once the victim was discovered.

n) Victim Identification: We found an average of 2.3 victims for each round (where the number of victims detectable by our sensors ranged over $4-5$ for each round). We deployed visual analysis only during the first contest, and audio sensors only during the finals. Using audio sensing, because of the noisy environment, we encountered a false-positive victim detection just in the last mission.

C. Experiences in our domestic arenas

We tested the control architecture and the interface effectiveness in our domestic arenas comparing three possible settings: i. *fully teleoperated*: navigation, slam, and vision disabled; ii. *supervisioned control*: the monitoring system was enabled and the operator could supervise the rover status and take the control whenever this was needed (mixed initiative control); iii. *autonomous control*. During the supervised control tests, we considered also the percentage of time spent by the operator in teleoperation 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 (average) 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

Concerning *global exploration*, the performance of the supervised 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 GM and NU tools 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 supervised 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).

VII. Conclusion

We presented a model-based approach to the execution and control of an rescue robotic system. We also showed how this model supports human-robot interaction during the rescue competition. In fact, the system improves the operator situation awareness providing a better perception of the mission status.

We briefly detailed some system's components in order to highlight the support given to the rescue operator and we reported some positive experimental results obtained during the last RoboCup rescue competition missions and missions performed in our domestic arena.

References

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

[2] S. Tadokoro, "Robocuprescue robot league," in *RoboCup-2002*, 2000, pp. 482–484.

[3] B. A. Maxwell, W. D. Smart, A. Jacoff, J. Casper, B. Weiss, J. Scholtz, H. A. Yanco, M. Micire, A. W. Stroupe, D. P. Stormont, and T. Lauwers, "2003 aaai robot competition and exhibition," *AI Magazine*, vol. 25, no. 2, pp. 68–80, 2004.

[4] A. Jacoff, E. Messina, and J. Evans, "A reference test course for urban search and rescue robots," in *FLAIRS Conference 2001*, 2001, pp. 499–503.

[5] J. Burke, R. Murphy, M. Coovert, , 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*, vol. 19, no. 1,2, pp. 21–38, 2004.

[6] R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 34, no. 2, pp. 138–153, 2004.

[7] D. J. Bruemmer, R. L. Boring, D. A. Few, J. L. Marble, and M. 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.

[8] S. Kiesler and P. Hinds, "Introduction to the special issue on human-robot interaction," *Special Issue of Human-Computer Interaction*, vol. 19, no. 1,2, pp. 1–8, 2004.

[9] H. Yanco and J. Drury, "A taxonomy for human-robot interaction," in *Proc. AAAI Fall Symposium on Human-Robot Interaction*, 2002, pp. 111–119.

[10] 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.

[11] B. K. Michael Baker, Robert Casey and H. 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".

[12] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, B. Chafin, W. Dias, and P. Maldague, "Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission," *Intelligent Systems, IEEE*, vol. 19, no. 1, pp. 8– 12, 2004.

[13] 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.

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

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

[16] A. K. Jonsson, P. H. Morris, N. Muscettola, K. Rajan, and B. D. Smith, "Planning in interplanetary space: Theory and practice," in *Artificial Intelligence Planning Systems*, 2000, pp. 177–186.

[17] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002, to appear.

[18] A. R. Cassandra and L. P. Kaelbling, "Exact and approximate algorithms for partially observable markov decision processes," Ph.D. dissertation, 1998.

[19] R. Bellman, M. L. Juncosa, and R. Kalaba, "Some numerical experiments using newton's method for nonlinear parabolic and elliptic boundary-value problems," *Commun. ACM*, vol. 4, no. 4, pp. 187–191, 1961.

[20] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[21] I. T. Nabney, *Netlab: algorithms for pattern recognition*. Springer, 2002.

[22] Y. M. and A. N., "Gaussian mixture model for human skin color and its application in image and video databases." in *In Proc. of the SPIE: Conference on Storage and Retrieval for Image and Video Databases (SPIE 99)*, vol. 3656, 1999, pp. 458–466.

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