Multi-objective Exploration and Search for Autonomous Rescue Robots

Daniele Calisi, Alessandro Farinelli, Luca Iocchi, and Daniele Nardi
Dipartimento Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria, 113 - 00198 Roma, Italy
e-mail: calisi@dis.uniroma1.it
e-mail: farinelli@dis.uniroma1.it
e-mail: iocchi@dis.uniroma1.it
e-mail: nardi@dis.uniroma1.it

Received 8 January 2007; accepted 23 July 2007

"Exploration and search" is a typical task for autonomous robots performing in rescue missions, specifically addressing the problem of exploring the environment and at the same time searching for interesting features within the environment. In this paper, we model this problem as a multi-objective exploration and search problem and present a prototype system, featuring a strategic level, which can be used to adapt the task of exploration and search to specific rescue missions. Specifically, we make use of high-level representation of the robot plans through a Petri Net formalism that allows representing in a coherent framework decisions, loops, interrupts due to unexpected events or action failures, concurrent actions, and action synchronization. While autonomous exploration has been investigated in the past, we specifically focus on the problem of searching interesting features in the environment during the map building process. We discuss performance evaluation of exploration and search strategies for rescue robots, by using an effective performance metric, and present evaluation of our system through a set of experiments.

1. INTRODUCTION

In recent years increasing attention has been devoted to rescue robotics, both from the research community and from rescue operators. Robots can consistently help human operators in dangerous tasks during rescue operations in several ways. Indeed, one of the main services that mobile robots can provide to rescue operators is to work as remote sensing devices reporting information from dangerous places that human operators cannot easily and/or safely reach.

A consistent part of rescue robotic research is focused on providing robots with high mobility capabilities and complex sensing devices. Such kinds of robots are usually designed to be teleoperated during the rescue mission and the knowledge about the mis-

Journal of Field Robotics 24(8/9), 763–777 (2007) © 2007 Wiley Periodicals, Inc.
Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/rob.20216

sion scenario is gathered by the human operator through the displayed output of the sensors (e.g., camera images). Performance evaluation for these kinds of rescue robots is consequently focused on the ability to drive the robot through terrains of measurable complexity [e.g., random stepfields (Jacoff & Messina 2006)], while taking into account operation constrains, such as lack of visibility of the robot in action, or control by single operator. Communication in this case must be guaranteed at all times, otherwise the robots become out of control.

Another branch of rescue robotics focuses instead on providing mobile bases with a certain degree of autonomy. Autonomous and semi-autonomous robots can process acquired data and build a high-level representation of the surrounding environment. In this mode, robots can act in the environment (e.g., navigate) through a limited interaction with the human operator. This also allows a human operator to easily control multiple robots by providing high level commands (e.g., "explore this area," "reach this point," etc.). Moreover, in case of temporary network breakdown, the mobile robotic platforms can continue the execution of the ongoing task and return to a predefined base position.

In this work, we consider autonomous exploration and search in an indoor unstructured environment. The goal is to explore and map the environment, while searching for mission relevant information that should be reported to the human operators. Such a search task is typically targeted towards detection and analysis of "interesting" features: human victims, temperature, presence of gas or any other substance, etc. Moreover, the relevant features should be located inside the environment, therefore the robot has to build a map of the explored area and localize itself inside the map. In the following, we refer to the above task as "exploration and search". The expected output for "exploration and search" is a 2D map of the planar environment, annotated with interesting features (e.g., victims). We also assume the environment to be static.

More specifically, we focus on multi-objective exploration and search. In fact, in rescue missions, heterogeneous sensing devices are often used for the map building system and for the victim detection system, and the two processes have different and, sometimes, conflicting goals. For example, the map building process must be accurate and fast: a laser can accurately acquire information from a long distance (up to 80 m). On the other hand, victim detection can be very demanding from a computational point of view and it requires a very short range of operation.

Autonomous exploration has been deeply investigated in the mobile robot literature (Latombe, 1991; Stachniss, Grisetti, & Burgard, 2005; Yamauchi, 1997; Jin, Liao, Polycarpou, & Minai, 2004). Most approaches, however, do not consider the problem of searching for interesting features inside the environment, while doing the exploration. On the other hand, several approaches have been proposed for searching the environment, using coverage techniques [see (Choset, 2001) for a survey] or aggregation-based techniques (DasGupta, Hespanha, Riehl, & Sontag, 2006). However, such approaches generally assume to operate in a known environment and do not address uncertainty in robot actions and perceptions.

The aim of the present paper is to present a novel approach to the multi-objective exploration and search problem based on an explicit high-level representation of the robot plans and to assess its performance through a systematic evaluation.

Our approach is based on a strategic level component, which is in charge of monitoring and driving the search and exploration process. Specifically, we model the strategic level using a formalism called Petri Net Plans (PNP) (Ziparo & Iocchi, 2006), which is based on Petri Nets (Peterson, 1981; Murata, 1989), a graphical modeling language for representing dynamic systems. The PNP formalism allows for an effective representation of concurrent processes, sensing actions, as well as interrupts and failures, which are needed to promptly react to new events and to handle the faults due to the high uncertainty of the information available. The introduction of a strategic level allows for an effective and flexible solution to the multi-objective exploration problem, as opposed to approaches based on the optimization of a weighted function [e.g., (Stachniss et al., 2005)].

The powerful formalism used for the strategic level allows specifying complex strategies that can address and resolve the conflicting goals of the different subsystems, while easily incorporating possible a priori knowledge about missions and environments into the system. For example, consider the case where a rescue system enters a building knowing that most of the people inside it were concentrated in one room (without knowing the exact map of the building and, therefore, without knowing how to reach such a room). The system should quickly explore the building as soon as a relevant number of people is de-

tected, then the mapping process should have less priority and the system should focus on precisely assessing the status of the found people. Such an exploration strategy is very hard to encode writing down a function to optimize, while it can be easily expressed using our strategic level.

The evaluation of the system is based on quantitative experiments in different operative scenarios using a Pioneer 3-AT mobile base equipped with a Laser Range Finder and a pan-tilt sensor unit including a stereo camera and a thermo sensor. A 3D simulator of Rescue Scenarios (USARSim[1]) has also been used in the experiments. The aim of the experiments is to show that the introduction of a strategic level allows for a high flexibility of the system in defining different strategies for rescue missions and it allows for an easy integration of available a priori knowledge about the operational scenario.

The metric used to evaluate the system takes into account at the same time the need for being easily applicable on real systems and the requirement of being totally objective (i.e., not requiring the definition of arbitrary weighting factors to account for different types of information to gather from the environment). Consequently, differently from other metrics (for example, those used in RoboCup Rescue competitions), we propose a metric based on a set of functions, each one measuring the ability of the system to discover information of a particular kind, without trying to merge these results into a single value by weighting the importance of each feature. This metric does not directly compare two systems/strategies when it is not evident that one outperforms the other, i.e., where the result would be situation dependent. In these cases, evaluation done by human operators seems best suited.

The paper is organized as follows. In the next section we present related work and compare our solution with previous work. Section 3 describes our proposal for multi-objective exploration and search; Section 4 describes our robotic system and implementation details about the software modules that implement mapping, localization, navigation, and victim detection. Section 5 describes the experimental setting and performance evaluation of the proposed system. Finally, Section 6 draws the conclusions.

[1]http://usarsim.sf.net

2. RELATED WORK

The exploration and search problem in rescue missions is the main focus of the RoboCup Rescue competitions,[2] and therefore all the participant teams have to deal with this problem and have implemented specific solutions. In case of tele-operated robots, the multi-objective exploration and search is solved by the human operator, who decides the activities of the robot. The teams that have implemented autonomous behaviors usually adopt a strategy that prefers victim analysis to map exploration, also according to the specific score system used in the competitions. Therefore, to the best of our knowledge, a formalization of the problem as a multi-objective search task and a systematic analysis of the developed solutions have not been presented. In addition, there are several works in the literature addressing exploration and search problems, but not considering some relevant issues encountered in rescue missions.

The exploration problem is usually seen as an optimization problem, in which the expected total time of the exploration or a related value has to be optimized (e.g., the information gain has to be maximized, the total travel cost minimized, etc.). Several approaches address the problem by defining utility functions for the choice of the next position to travel to. In these utility functions, one has to consider both the costs of the actions and their information gains. Typically, one can take into account the minimization of the total time along with other features, such as map precision, and include them in the utility function. For example, Yamauchi (Yamauchi, 1997) proposes a frontier based exploration, where the robot chooses the next point to reach based on its distance from the available frontiers. A frontier is a part of the environment, which divides explored from unexplored space. Following such an approach, the robot tries to minimize the total exploration time. González-Baños and Latombe in (González-Baños & Latombe, 2002) propose an extension of the next best view (NBV) method applied to the exploration task. The proposed approach extends previous techniques [e.g., (Pito, 1996)] developed for computer vision, considering issues specific to mobile robots (e.g., localization and obstacle avoidance).

While the task of exploring an unknown environment with the goal of building a map can be suitably

[2]www.rescuesystem.org/robocuprescue/

modelled as an optimization problem, as presented above, exploration and search usually entails a multi-objective optimization (Coello, 1999). Several approaches address the problem by defining a single utility function, which depends on several parameters to optimize. For example, Stachniss et al. (Stachniss et al., 2005) propose an approach that combines mapping, localization, and exploration. Their method evaluates the cost and information gain of the actions that the robot has to perform and combines them with a weighted function to be maximized. The work by Mathews and Durrant-Whyte (Mathews & Durrant-Whyte, 2006) addresses a similar problem. The authors focus on the control of multiple platforms involved in an information gathering task. The goal of the approach is to find the optimal joint actions for the mobile platforms in order to minimize the entropy of the probability distribution over the system state. The authors apply their approach to a distributed indoor mapping scenario. In both the latter two approaches, the actions are addressed mostly for the construction of a precise and accurate map and do not consider other kinds of interesting features to be analyzed and reported on the map.

On the other hand, Amigoni and Gallo (Amigoni & Gallo, 2005) address the multi-objective nature of the exploration problem, by considering a Pareto efficient approach: instead of finding an optimal solution for a weighted function, they suggest to find a solution that is good enough for all the measures to be optimized.

Also, the work by Jin et al. (Jin et al., 2004) explicitly addresses the trade-off between search and response to interesting targets. The work focuses on a team of UAVs searching for moving targets. Some of the targets are known beforehand and others appear dynamically. UAVs should thus search for possible new targets, while at the same time act in response to already found ones. This trade-off is addressed using a hybrid algorithm that switches between an instantaneous task assignment technique, which considers only the current world state in the task assignment process, and a predictive task assignment technique, which tries to predict future states of the targets. The authors show that the hybrid algorithm successfully addresses the mentioned trade-off; however, the approach seems to be strongly dependent on the reference scenario. In particular, the determination of the threshold for switching between the two task assignment techniques is performed using an heuristic based on the results obtained in previous simulations.

The problem of searching an object for a searcher with limited perceptions, but with a priori probability distribution of the target position, is addressed by DasGupta et al. (DasGupta et al., 2006). This problem amounts to defining a path that maximizes the probability of finding the target. The authors use an aggregation-based technique to approximate the problem and solve it in polynomial time. However, the results apply only to a scenario where the map is given beforehand, and a probability function of the target position is known.

All the approaches mentioned above implement multi-objective exploration either as a numeric optimization process or with a task-specific solution. The use of an explicit high-level representation of the information and of the plan to be executed has not been addressed.

On the other hand, multi-objective optimization can be addressed by making use of Markov Decision Processes (MDP, POMDP) (Pineau & Gordon, 2005). Markov Decision Processes allow for high-level representation of actions and strategies and for devising optimal policies (i.e., sequences of actions) to be executed, maximizing a reward function defined over the domain. Such approaches, however, are highly computationally demanding, especially for complex environments, where several actions and states have to be modeled. These approaches can address multi-objective problems only by including each objective reward (using weighting factors) in the global reward function. Moreover, in this paper, we are not interested in determining optimal policies, but in providing a flexible and efficient way of implementing exploration and search strategies.

In this paper, we adopt an approach to multi-objective exploration based on an explicit high-level representation of the strategy to be implemented during the mission. The use of Petri Net Plans allows for an expressive representation language and an efficient implementation. In addition, the ability of representing in a qualitative way the knowledge acquired during the mission, the decision conditions that are used to implement a multi-objective strategy, and a priori knowledge about the rescue mission is fundamental for an effective implementation of autonomous rescue robots. Such a high-level representation has the following advantages over a numerical optimization technique: (1) it does not require the

definition of empirical numerical values to guide the exploration and search task; (2) it allows for an easy characterization of the strategy to be adopted in a given scenario; (3) it allows for an easy injection of a priori knowledge about the mission, which is usually given in qualitative terms.

3. HIGH-LEVEL REPRESENTATION OF MULTI-OBJECTIVE EXPLORATION AND SEARCH STRATEGIES

Autonomous robots performing exploration and search tasks must be able not only to build a map of the environment and to localize themselves with respect to it (SLAM), but also to decide the sequences of movements needed to explore, in an optimal way, the whole environment.

It is useful to characterize exploration as a next-best-view (NBV) problem (González-Baños & Latombe, 2002), i.e., computing a sequence of sensing positions based on the data acquired at previous locations. An optimal NBV algorithm moves the robot to positions, where it can achieve the maximum information gain (given the current knowledge of the environment). The algorithm structure is given by the following steps:

1. select or choose the next view point;
2. navigate or move the sensor to that view point;
3. acquire information from the sensor;
4. integrate this information with the global knowledge of the object/environment.

This process does not take into account multi-objective searches. When the exploration task and the search tasks are performed simultaneously, it is often convenient to interrupt the action to reach a given target if a better (e.g., more promising, closer) one is detected during navigation. For example, in a rescue mission, it is possible to notice a victim while reaching the next frontier for exploration and map building. In this case, we would like the robot to get information about the victim as soon as possible, possibly before reaching the current frontier target. The NBV strategy, as stated above, does not implement such behavior, since decisions are taken only when the current target has been reached.

Moreover, the obvious approach of reconsidering the choice of the next target at each cycle (or with very high frequency) can be unfeasible due to the computation cost and the uncertainty in robot perception that may cause instability of behavior and consequent loss of performance. Moreover, one needs to face unexpected navigation failures that, in a rescue mission, can often happen.

In this section we present a highly flexible mechanism for realizing the decision level of an autonomous robot involved in a multi-objective exploration and search task. In particular, we describe an approach that provides a very flexible way to decide (1) when it is necessary to reconsider decisions and (2) how to choose among different targets.

The basic structure of our exploration and search strategy is inspired by the general NBV algorithm structure and is divided into two steps: (1) detection, evaluation, and selection of target candidates and (2) navigation towards the chosen target. However, effective implementations of complex exploration and search strategies are achieved by exploiting the high flexibility of the high-level strategic component that allows for explicitly modelling interrupt conditions and reactions to navigation failures.

To this end, we have chosen to explicitly model the strategic level, by developing a Plan Executor Module, based on Petri Net Plans. Petri Net Plans exploit Petri Net ability of representing concurrent execution of discrete state systems and are thus more powerful representations based on finite state machines. In particular, PNP allows for the representation of different kinds of actions (ordinary and sensing actions) as well as many standard control structures: conditional execution, loops, concurrent execution, interrupts, communication and synchronization actions for multi-agent systems, etc.

The use of PNP provides the developer with a simple to use high-level formalism for defining complex behaviors, with graphical tools for writing and debugging plans, with verification tools to check consistency of the plans with respect to the intended behaviors, and with simulations of plan execution that are used to test its correct behavior. These are important advantages with respect to hard coding the robot behaviors, which speeds up development time, increasing implementation correctness and system reliability.

While a detailed presentation of the Petri Net Plan formalism is outside the scope of the present paper [see (Ziparo & Iocchi, 2006) for details], we provide intuitions of the PNP structures and we present

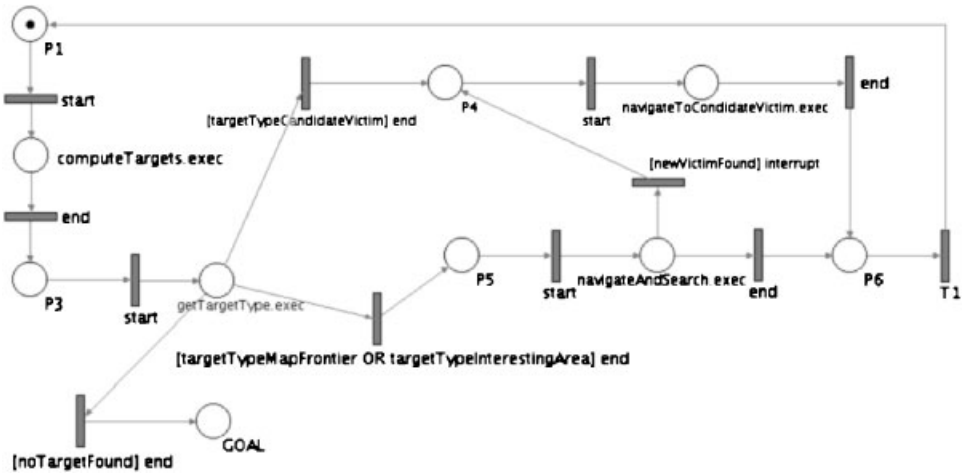

Figure 1. One of the plans used in the experiments, called "Victims First."

examples of multi-objective exploration and search strategies developed with such a formalism.

One instance of such plans is depicted in Figure 1. All the actions begin with a start transition, have an execution state exec, and terminate with an end, a fail, or an interrupt transition. The token contained in the place P1 indicates the initial marking, representing the beginning of the execution. This token is then propagated in the network places as specified by the transitions, which can be fired when the input places are *active*, i.e., when they contain at least one token, and the attached conditions (written between square brackets) are true.

The plan begins with a computeTargets action. This action performs three activities: (1) it computes map frontiers, (2) it evaluates candidate features to be analyzed, and (3) it chooses one of them as the next target position. The next action, getTargetType, is used as a conditional construct to guide the plan according to the target type. If it has been chosen to go to a possible victim, the subplan navigateToCandidateVictim is performed, otherwise the subplan navigateAndSearch is executed. In this plan, an interrupt transition is present in the execution step of navigateAndSearch. This is controlled by the condition newVictimFound and it is used to switch from the navigateAndSearch behavior to the navigateToCandidateVictim one. In other words, it allows for implementing a strategy where victim analysis is preferred to map exploration.

The subplan navigateToCandidateVictim, shown in Figure 2, starts two parallel actions, pantiltSearch and navigate, using the fork operator before their activation. While the robot navigates to the chosen target, the pan-tilt unit will move searching for new interesting places (using analysis of data coming from the thermo and stereo sensors, which is explained in the next sections). When the robot approaches the victim, the pantiltSearch action is stopped and the stereo sensor is pointed towards the candidate victim (action pantiltPoint). When the target position is reached, both navigate and pantiltPoint actions are stopped: this allows for a more stable acquisition of data and for using all the computational power required by the complex analysis of 3D points extracted with the stereo sensor (action analyzeVictim). The subplan terminates either after the victim analysis action or in the presence of navigation failures. In this latter case, action synchronization is used to interrupt also the pan-tilt search action. Notice that, when this subplan terminates, the main plan will loop to compute a new target and go to it.

4. SYSTEM DESCRIPTION

In this section, we briefly explain some implementation details of our robotic system for autonomous search and exploration. Our system has been validated through a set of experiments performed both in simulation and in real rescue arenas. It is worth pointing out that we use the same software modules and

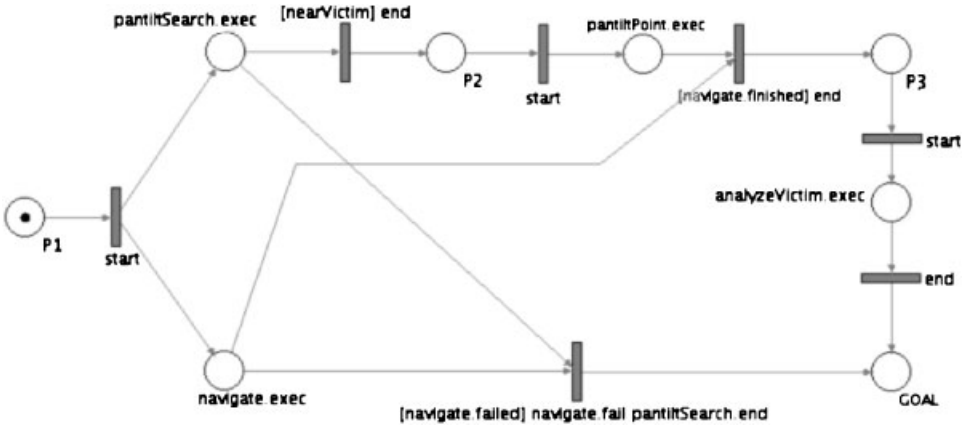

Figure 2. The subplan navigateToCandidateVictim.

the same strategic plans both in the simulated and in the real scenarios. We modeled our world in the simulated environment to be identical to the real one; in this way, the comparison between the behavior in both environments is straightforward. In the following subsections we provide an overview of the system and the components that are necessary to perform a search and rescue mission. Moreover, we detail important issues regarding the modeling of our rescue scenario in USARSim.

4.1. Robotic System

Our robot is a wheeled mobile robotic base, Pioneer 3-AT, equipped with a laser range finder. An onboard computational unit is responsible to run all the software modules needed by the robot. Since moving the robot can be very time consuming, the sensors used for feature detection (a stereo-camera and a nontouch thermo sensor) are mounted on a pan-tilt unit. In this way, while the robot is moving, these sensors can point towards targets that are not in front of the robot.

The possible targets to be explored are computed using three different algorithms, which aim at reaching three different (and concurrent) goals:

- The first goal is to build the map of the environment using laser range finder data; therefore, the first algorithm computes frontiers between free explored space and unknown portions of the map using a wavefront expansionlike algorithm (Latombe, 1991). The algorithm allows to compute frontiers incrementally; as pointed out in (Yamauchi, 1997), since frontiers move, while the robot acquires new information about the environment, the method will eventually guide the robot to explore all the accessible environment (though there is a possible Zeno-like Paradox, that prevents the map to be fully explored, this is very unlikely to happen).
- The second goal is to find "interesting" features in the environment: in our implementation, we assume that the interesting features are always intercepted by a laser scan. Therefore, we need to search for victims only along the obstacles detected by the laser range finder, thus avoiding unnecessary search in open spaces.
- A search algorithm scans the areas given by the method above and returns a list of possible features. These need to be analyzed in order to be confirmed or declared false positives. This analysis is computationally intensive and requires the robot to stand still: this list of possible features represents the third set of candidate targets.

In addition, the search and exploration subsystem relies on other modules, which are common among robotic systems and represent the low-level components that allow the robot to realize the decisions taken by the strategic level. These modules are

integrated in a framework that enables concurrent processes and communication among them [details can be found in (Farinelli, Grisetti, & Iocchi, 2006)]. In the following we provide a brief description of the most important software modules.

- The **SLAM Module** deals with map building and localization at the same time. We use a scan matching algorithm for on-line localization in conjunction with an occupancy grid for map building. The map built using such an approach is locally consistent and the error accumulated is small enough to allow for safe navigation, good quality maps, and quite precise victim localization.
- The **Navigation Module** enables the robot to reach a target position; a coarse "topological" path-planner computes a path and a lower-level module is in charge of maneuvering the robot to follow the provided path [see (Calisi, Farinelli, Iocchi, & Nardi, 2005) for details]. The lower level module uses two modalities: an accurate motion planner based on Randomized Kinodynamic Trees [see (LaValle & Kuffner, 1999)] to maneuver the robot in cluttered space, and a fast reactive navigation module based on the Vector Field Histogram that is able to steer the robot at high speed when far from obstacles.
- The **Human Body Detection Module** (HBD) performs two processing steps. The first step is fast, but not accurate, and is used to identify interesting places where a human body could be located. The main sensor used in this step is a nontouch temperature sensor. The second step uses temperature information, stereo vision, and a human body model database [see (Bahadori Ghouchani, 2006) for details]. It is very computationally intensive and slow, but provides good accuracy. For this reason, it should be activated only in those areas that are declared interesting in the first step; moreover, its range of operation is short (between 0.5 and 1.5 m).

4.2. Simulated Robotic System

The experiments in the simulated scenario are performed using USARSim, a 3D simulation environment based on a game engine. This framework allows for a realistic modeling of robots, sensors, and actuators, as well as complex, unstructured, dynamic environments. Recently, it has been used to measure the performance of robotic platforms and multi-robot systems in USAR (Urban Search And Rescue) scenarios (Wang, Lewis, & Gennari, 2003; Balakirsky, Scrapper, Carpin, & Lewis, 2006).

In USARSim, we can use exactly the same software modules used in the real robot (SLAM, Navigation, etc.) by replacing the interfaces to the real sensors (drivers), with interfaces to the simulator.

The simulation environment provides models of our mobile base and of the pan-tilt unit. It also includes models of laser range finders, odometry, and sonar sensors. The only missing device is the nontouch temperature sensor. However, we are not interested in using the simulator to test and evaluate the HBD system; we rather want the simulated system to have similar behaviors with respect to the real robot when executing the exploration process. To this end, in the simulator, we have used RFID tags located near victims and put an RFID reader on the robot. When the RFID tag is detected by the reader, the robot receives data similar to the ones provided by the thermo sensor on the real robot, i.e., a position in the environment were a possible victim was detected. For the second step (i.e., victim confirmation), we replace it with the request to the simulated robot of approaching the victim and taking a snapshot of the scene.

5. EXPERIMENTS

A set of experiments has been performed in order to show and analyze the effectiveness of using the high-level strategic module. More specifically, we have evaluated the ability of introducing some qualitative a priori knowledge about the mission in the system. In the following of this section, we first present the performance metric used to evaluate the different strategies and then show the results in four different scenarios.

5.1. Performance Metrics

An important aspect of the search and rescue tasks described in this article is the definition of effective metrics for measuring performance of autonomous robotic systems. The goal is to choose the actions that gain the maximum information in the minimum

time (or minimum energy), and performance metrics can be divided in two groups: measure of time (or energy) needed to acquire all the information from the environment and measure of information acquired with limited (time or energy) resources.

The first group considers a complete discovery of all the information from the environment and measures the time (or energy) needed to accomplish such a task. This is a good and objective metric, but it requires that the task is completely finished, and cannot be used to evaluate partial execution of the task. Notice that in many cases a complete discovery of the information from the environment is not feasible, due to time or resource constraints in large environments (e.g., battery life), or to the inability of the robot to detect some features.

Another set of metrics determines a bound on the resources available to the robot and measures performance of the search task once the limit has been reached. This bound can be given both in terms of time and in the form of some kind of energy consumed by the robot (that can be approximated by limiting the total path the robot can drive along). This metric is easily applicable to real systems, since it takes into account intrinsic limitations on system resources, but it requires measuring how much information has been gathered from the environment at a given point. This is not easy in the presence of multiple objectives (e.g., map to be explored and victims to be found), and in many cases [see, for example, the evaluation function used within the RoboCup Rescue competitions (Jacoff, Weiss, & Messina, 2003)[3]] the introduction of weighting factors that make the measure not objective is needed.

In order to use an objective performance metric that is applicable to real robotic systems, in the following we will consider several functions, one for each feature that must be discovered and measured from the environment.

More specifically, our performance metric is given by a set of functions

$$V \equiv \{ v_i(t) | i = 1, \ldots, n \}, \tag{1}$$

one for each feature i to be measured from the environment. Each function $v_i(t) \rightarrow [0,1]$ measures the percentage of information related to feature i measured at time t. For example, in the case of map building, $v_i(t)$ is the amount of explored map with respect to the total size of the environment, while in the case of victim detection it is the percentage accuracy in determining the number of discovered victims versus the total number of victims that are in the environment. More specifically, for victim detection the evaluation function is

$$v_i(t) = \mathbf{max} \left(0, 1 - \frac{|\mathbf{found\_victims} - \mathbf{total\_victims}|}{\mathbf{total\_victims}} \right),$$

where **found_victims** refers both to candidate and confirmed victims. This function guarantees that v_i is always between 0 and 1 even in the presence of false positives.

This metric has the advantage of being easily applicable to real systems; moreover, it does not require the definition of weighting factors. However, it does not allow for a direct comparison between different strategies or different systems, in cases where all the single functions of one system/strategy do not outperform the ones of the other.

5.2. Experimental Scenarios

Experiments have been performed in four different scenarios that require different mission strategies to accomplish the goal. In the following subsections we describe each scenario and the particular strategy implemented for it. Moreover, we discuss the performance of our approach with respect to the particular goal of the mission. All the experiments have been performed in a limited time (10 min) and we measured the values of information $v_i(t)$ for the interesting features: $v_1(t)$ measures the explored map, $v_2(t)$ the candidate victims, $v_3(t)$ the confirmed victims, $v_4(t)$ (used only in Scenario 2) the fire detection.

In Scenarios 2–4, qualitative a priori knowledge about the mission is provided. This is taken into account in our approach by modifying the high-level Petri Net Plan in order to better fit the desired behavior.

In Scenarios 1 and 2, the implemented strategies are compared with a greedy policy (we call it "Simplest") that uses a utility function to decide the next position to explore. This strategy can interrupt the navigation to the current goal if another candidate target, with a higher utility value, has been found.

[3]http://www.rescuesystem.org/robocuprescue/or
http://www.isd.mel.nist.gov/projects/USAR/competitions.htm

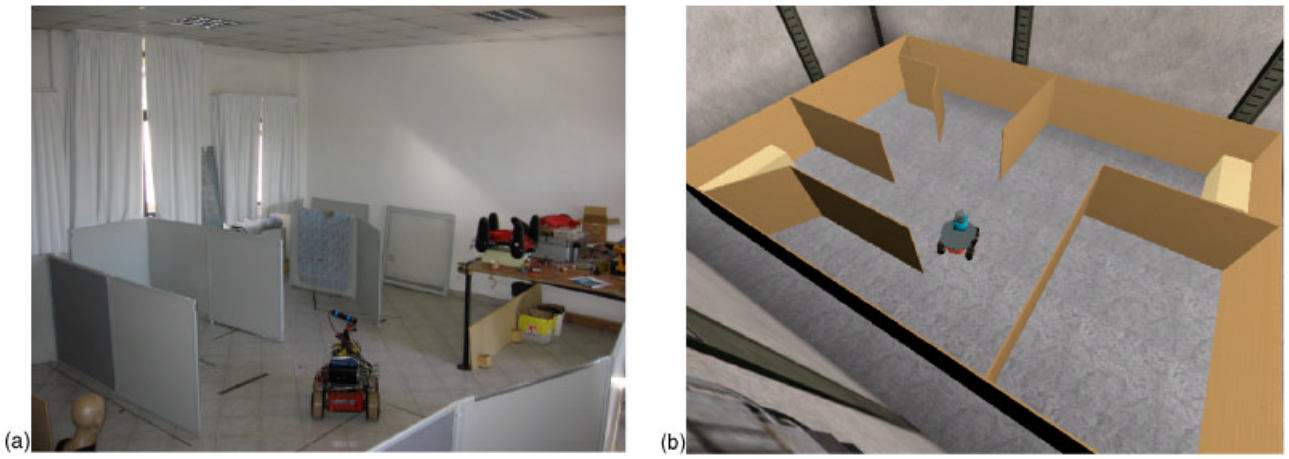

Figure 3. The robot in the simulated and real arenas used for the experiments in Scenarios 1 and 2.

The utility function computes the path distance to the candidate targets and then multiplies this value by a fixed *weighting coefficient* that characterizes the importance of the target type.

Scenarios 3 and 4 are more complex, and the a priori knowledge cannot be simply expressed by defining a function to optimize. Therefore, in these scenarios, we show how the high-level representation of the robot plan can be suitably modified in order to take into account complex qualitative knowledge.

The experiments have been performed using both our rescue robot operating in a test bed arena and a simulated robot in USARSim. More specifically, the experiments of Scenarios 1 and 2 have been performed both in the real and in the simulated environment, while the ones in Scenarios 3 and 4 have been performed only in the simulation. For each experiment, ten runs have been performed (both with the real robot and in the simulator) starting from the same initial configuration of the environment and of the robot. Results reported below are averaged over these ten runs.

As described below, experiments in simulated scenarios are performed using two different maps (one for Scenarios 1 and 2 and one for Scenarios 3 and 4) with two different robot starting poses and victim configurations for each of them.[4]

[4]Simulated scenarios, robot configurations, and results are available for experimental comparison of other approaches.

5.2.1. Scenario 1

In the first scenario, the goal is to find as many victims as possible with no a priori knowledge on the environment. For this scenario we have performed experiments in the virtual and in the real environments, aiming at assessing the basic behaviors of the system. The arena is built following the RoboCup Rescue Yellow Arenas specifications: it is planar and has a mazelike structure. The map used in the real and the simulated environment is the same and it is shown in Figure 3. In both cases there are three victims in an area of 7×5 m^2. The robot goal is thus to find and confirm all potential victims. The plan that determines the strategy for this scenario is the one already described in Section 3: the robot stops the exploration as soon as a new potential victim has been found and then it moves to a location suitable for the confirmation step. We call this strategy "Victims First."

The results of the experiments are shown in Figure 4, where we report, for each time step, the number of victims detected (both potential and confirmed), the number of victims confirmed, and the portion of the total map explored. As already mentioned, the reported values are averaged over a set of ten experiments. The left graph is relative to experiments in the real scenario and it can be compared with the right one that shows the results in the simulated environment. The results of the two sets of experiments show a similar trend of the evaluation functions, indicating a consistent behavior in

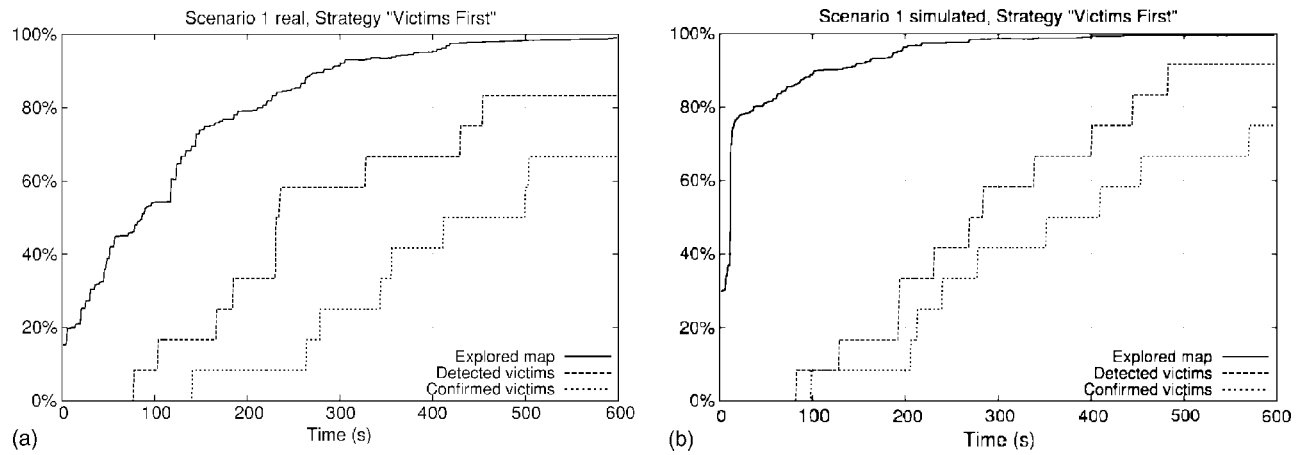

Figure 4. Results of the experiments in Scenario 1: (left) results in the real scenario using the "Victim First" strategy, (right) results of the same strategy in the simulated scenario.

the two environments. In fact, it is possible to see that victims are analyzed as soon as they are discovered. Notice that, while the trend of the functions is similar, their values present some differences that are due to the different behavior of the simulator with respect to the real environment.

This strategy has been also compared with the "Simplest" one, where the importance factor to confirm a candidate victim is much higher than the others (i.e., exploring other part of the environment and finding other candidate victims). The two strategies have the same behavior, hence the graphs are very similar to the above ones.

5.2.2. Scenario 2

In this scenario we introduce a new feature that represents the source of the incident (it can be the source of a fire, a chemical agent, etc.). There is only one such feature and the priority of discovering it is considered higher than finding the victims, since it may still represent an active threat. For the simulated experiments, we used a particular kind of RFID to model this feature, while in the real experiments a heat source with a very high temperature (about 100 °C) is used. In the real system, fire detection is performed by the same software module that is used for detecting potential victims, by using a different interpretation of data coming from the thermo sensor. In both cases, we assumed that a further detection step to confirm this observation is not needed.

The strategy implemented for this scenario (called "Fire First") is a slight variation of the one mentioned in the previous subsection. We want the robot to find the threat as soon as possible, avoiding the time-consuming analysis of candidate victims during the search. Therefore, potential victims are not considered when computing the next target to explore or search (in the computeTargets action of Figure 1), though we may still find new potential victims. As soon as the fire has been found, the priority of the strategy changes and, afterwards, the robot can focus on analyzing potential victims (i.e., those previously found and new ones as soon as they are detected).

To this end, we changed the computeTargets action by transforming it in a subplan that is shown in Figure 5. Here a conditional action checks if the fire has already been found and selects one of two different instances of the computeTargets action: the first considers victims to be analyzed, while the second does not.

The "Simplest" strategy for this scenario has been configured by ordering the importance factors of the features as follows: detecting fire has higher priority than confirming victims and exploration. The results of the set of experiments in the real environment are shown in Figure 6. The ones from the simulated environment have a similar trend and are thus omitted. The event of the discovery of the fire is also included. In the single experiment, this value can be 0 (fire not found) or 1 (fire found), but since

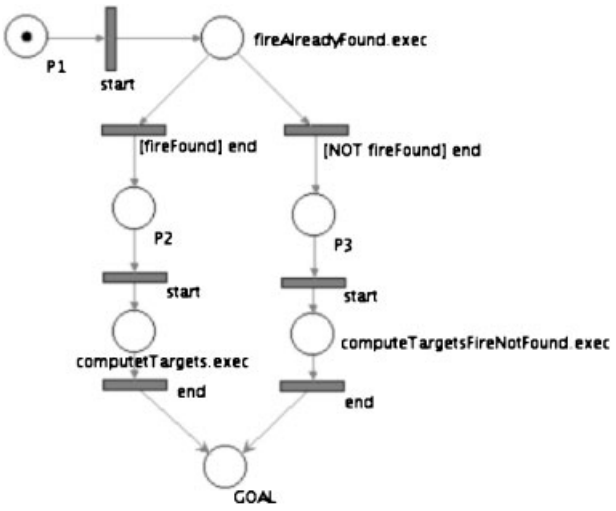

Figure 5. The subplan computeTargets, used in Strategy "Fire First."

the graph shows the average over ten experiments, intermediate values are present (for each "step" of the function measuring fire discovery, we know that at least one experiment found the fire at that time).

The results of the experiments show that the "Fire First" strategy handles effectively the priority of the fire search: the robot never stops the navigation to perform the second time-consuming step of the human body detection algorithm, before the fire discovery. The main difference with respect to the "Simplest" strategy is that, since it uses a fixed utility function for the choice of the next target to explore, the system behaves like it is maintaining the search of the threat as the primary goal of the mission, even after having found the threat. In other words, with the "Simplest" strategy, the second step of the victim analysis (and the consequent confirmation) can be performed only when the whole environment has been explored and all possible threats and victims found, while this process can be initiated immediately after the fire discovery by the "Fire First" strategy. The outcome of this behavior is that the "Fire First" strategy confirms a higher number of victims. In this scenario, we can notice that a dynamic reallocation of the weights of the "Simplest" strategy will result in a behavior similar to that of the "Fire First." Anyway, this solution does not allow for the flexibility of using the high-level representation.

5.2.3. Scenario 3

The goal here is to find as many victims as possible, with a priori knowledge of their location. This can be represented, for example, by a probability distribution over a metrical map or by some topological high-level statement (e.g., victims are mostly in the third room on the left). Although we have such information, we do not yet know the (metric) map of the environment, which has to be built on-line.

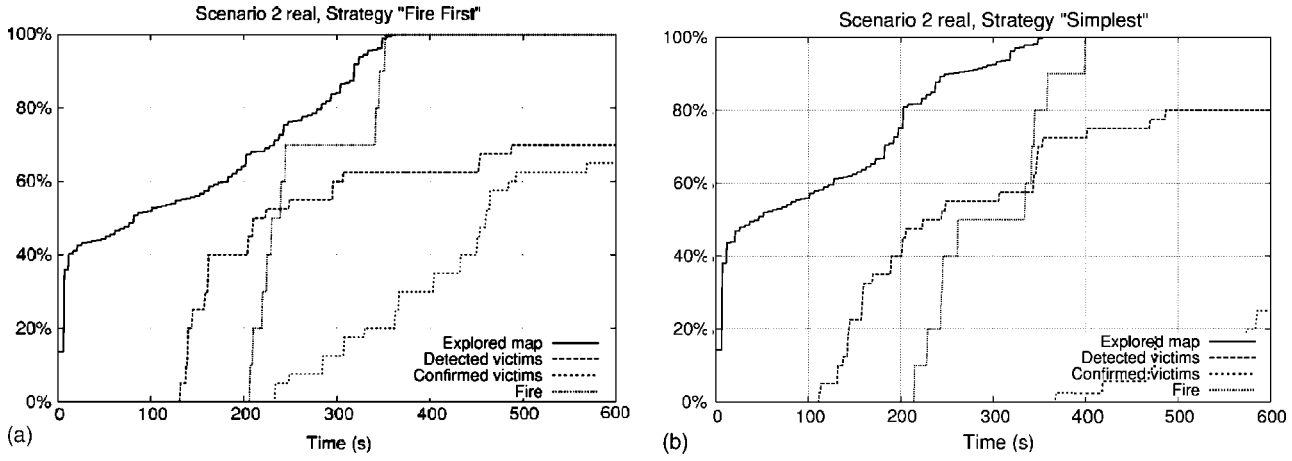

Figure 6. Results of the experiments in Scenario 2: on the left the strategy "Fire First," on the right the "Simplest" strategy, both in real environments.

Figure 7. The officelike environment used for Scenarios 3 and 4.

The plan-based strategy here is another variation of the "Victims First" strategy, where the action computeTargets takes into account the probability distribution in the decision of the next target to explore, giving more chances to candidate targets that are in areas where the probability distribution is higher. We call this strategy "Biased."

Instead, the "Simplest" strategy cannot be used as it is; it should be rewritten in order to take into account the notion of probability distribution.

The simulated scenario represents an officelike environment (see Figure 7), with a long corridor and five rooms. The a priori knowledge gives a higher probability to find victims in one of the rooms and the robot should concentrate its search in that room, although it is possible to find other victims elsewhere. In particular, we put four victims inside a room and one in the corridor.

Results of this experiment performed in the simulated environment are shown in Figure 8. In this figure, the results are compared with the outcome of the "Victims First" strategy in the same environment, in order to show how the a priori knowledge about the victim location leads to a consistent performance improvement, i.e., almost all the victims are confirmed by exploring only a portion of the environment.

5.2.4. Scenario 4

In the last scenario, the a priori information about the mission is the knowledge that victims are probably grouped in clusters. However, we do not know anything about their locations in the map. The implemented strategy, called "Go Away from Confirmed" follows the usual structure of the other experiments, except that, when a victim is found and confirmed, the computeTarget action will not consider other potential victims that are in the same area. This allows searching for other clusters as soon as a victim (that can be considered as the representative of its cluster) has been detected and confirmed.

In this scenario, we use a map similar to that of Scenario 3, in which we put five victims in one room and another far away from them, so that we have two clusters of victims.

Results in Figure 9 show that the strategy pushes the robot to explore unknown areas as soon

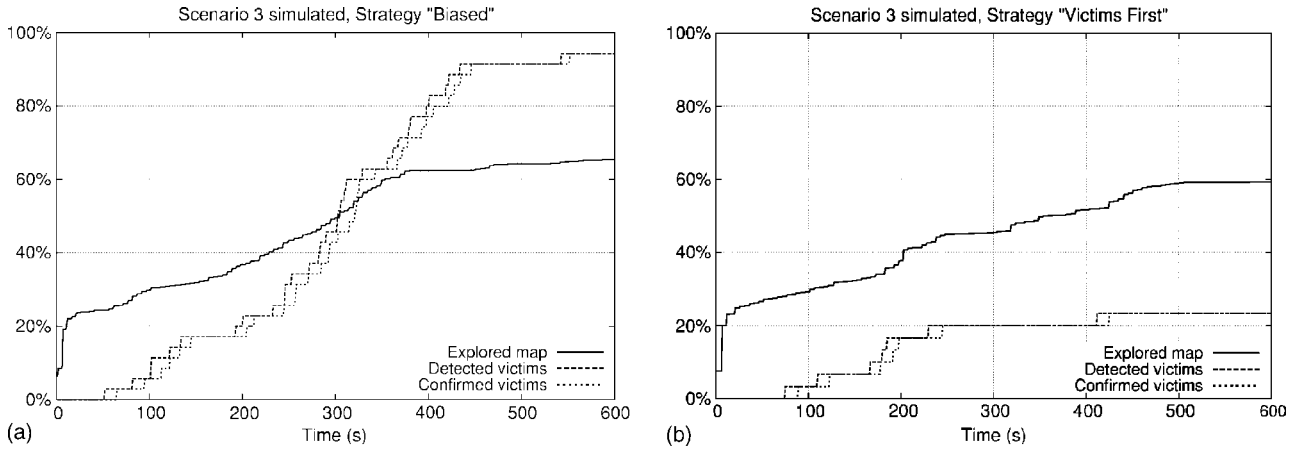

Figure 8. Results of the experiments in Scenario 3: on the left the strategy "Biased," on the right the results of the "Victims First."

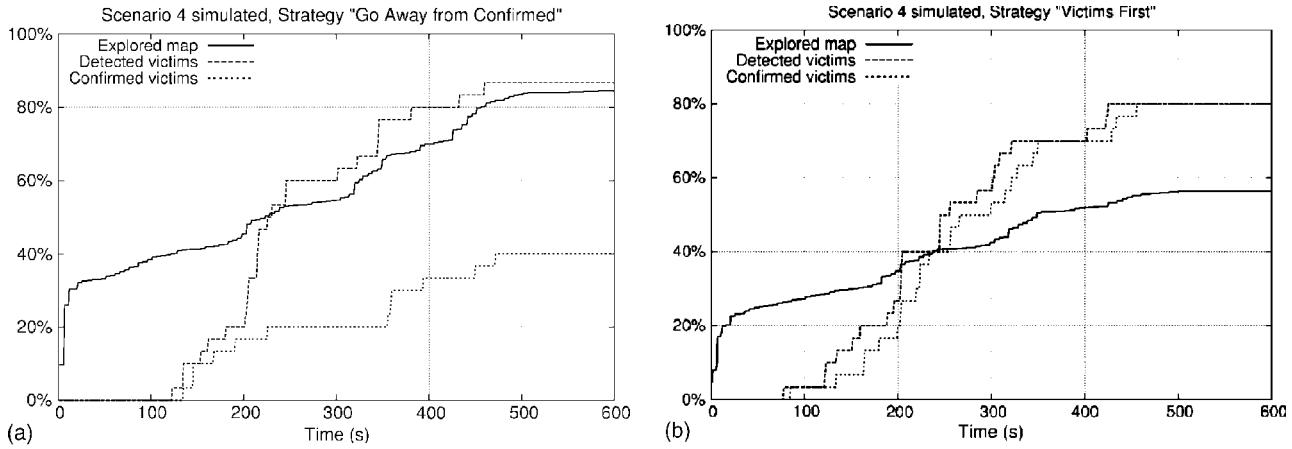

Figure 9. Results of the experiments in Scenario 4: on the left the strategy "Go Away from Confirmed," on the right the results of the "Victims First."

as it detects and confirms the first victim (remember that since there are five victims, 20% means one victim). In this case, we are not interested in the number of confirmed victims, but in finding and confirming both the clusters of victims in the room (by finding and confirming one of them) and the victim in the corridor, isolated from the rest. To show the correct behavior of this strategy, we show also the results of the "Victims First" strategy, which, following correctly its goal, will confirm as many victims as possible. Since the room with the four victims is nearer than the isolated victim, this strategy discovers and confirms, in all experiments, the four victims in the room, but has not enough time to discover the second cluster.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have investigated multi-objective exploration and search in a rescue environment. Specifically, we focused our attention on autonomous robots that can accomplish missions without direct control from the operator. We have described a solution based on a high-level framework for representing plans. In this way, it is possible to implement exploration and search strategies that depend on the features of the operational scenario and on the objectives of the mission and can be effectively applied by the rescue robot.

Different experimental scenarios have been devised to suitably evaluate the performance of exploration and search strategies. Moreover, a comparison of the proposed approach with methods based on utility function optimization has been performed. The reported results show that our formalism allows for easy implementations of different strategies that are effective in the considered rescue scenarios. Moreover, based on the experimental analysis, we argue that a proper evaluation of the performance in multi-objective search and exploration requires user oriented criteria to assess the effectiveness of the proposed methods.

As future work the approach could be extended by addressing more complex scenarios, where some of the assumptions made in the present work are not applicable: for example, dynamic scenarios where events occur during the mission (e.g., doors are opened/closed), 3D environments (e.g., multi-level environments), presence of moving victims, and additional features to be measured from the environment. In our opinion, more complex scenarios should further emphasize the features of the proposed approach.

Finally, the possibility to deploy teams of robots, which seems particularly attractive from a practical viewpoint, cannot simply be handled by summing the results achieved individually by the robots. Both the quality and the correctness of the information depend on how results are combined. Specifically, the overall result of search and exploration is clearly dependent on whether and how the maps built by the

individual robots are merged. Moreover, the search and exploration strategy is also influenced by the cooperation method adopted by the team: the analysis of the experiments and the use of more refined evaluation metrics with teams of robots must therefore include an additional dimension to account for cooperation.

REFERENCES

Amigoni, F., & Gallo, A. (2005). A multi-objective exploration strategy for mobile robots. In Proceedings of the 2005 IEEE International Conference on Robotic and Automation, Barcelona, Spain.

Bahadori Ghouchani, S. (2006). Human body detection in search and rescue missions. Ph.D. thesis, University of Rome 'La Sapienza', Dipartimento Di Informatica e Sistemistica.

Balakirsky, S., Scrapper, C., Carpin, S., & Lewis, M. (2006). USARSim: providing a framework for multi-robot performance evaluation. In Proceedings of the International Workshop on Performance Metrics for Intelligent Systems (PerMIS) .

Calisi, D., Farinelli, A., Iocchi, L., & Nardi, D. (2005). Autonomous navigation and exploration in a rescue environment. In Proceedings of the 2nd European Conference on Mobile Robotics (ECMR), pp. 110–115, Edizioni Simple s.r.l., Macerata, Italy.

Choset, H. (2001). Coverage for robotics—a survey on recent results. Annals of Mathematics and Artificial Intelligence, 31, 113–126.

Coello, C. A. C. (1999). An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, & A. Zalzala (eds.), Proceedings of the Congress on Evolutionary Computation, vol. 1, pp. 3–13, Washington DC. New York: IEEE Press.

DasGupta, B., Hespanha, J.P., Riehl, J., & Sontag, E. (2006). Honey-pot constrained searching with local sensory information. Journal of Nonlinear Analysis: Hybrid Systems and Applications, 65(9), 1773–1793.

Farinelli, A., Grisetti, G., & Iocchi, L. (2006). Design and implementation of modular software for programming mobile robots. International Journal of Advanced Robotic Systems, 3(1), 37–42.

González-Baños, H.H., & Latombe, J.-C. (2002). Navigation strategies for exploring indoor environments. Int. J. Robotic Res., 21(10–11), 829–848.

Jacoff, A., & Messina, E. (2006). DHS/NIST response robot evaluation exercises. In IEEE International Workshop on Safety Security and Rescue Robots, Gaithersburg, MD.

Jacoff, A., Weiss, B., & Messina, E. (2003). Evolution of a performance metric for urban search and rescue robots. In Proceedings of Performance Metrics for Intelligent Systems (PerMIS) Workshop, Gaithersburg, MD.

Jin, Y., Liao, Y., Polycarpou, M.M., & Minai, A.A. (2004). Balancing search and target response in cooperative uav teams. In Proc. of 43rd IEEE Conference on Decision and Control, pp. 2923–2928.

Latombe, J. C. (1991). Robot motion planning. Dordrecht: Kluwer Academic Publishers.

LaValle, S., & Kuffner, J. (1999). Randomized kinodynamic planning. In Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 473–179.

Mathews, G.M., & Durrant-White, H.F. (2006). Scalable decentralised control for multi-platform reconnaissance and information gathering tasks. In Proceedings of 9th International Conference on Information Fusion, pp. 1–8.

Murata, T. (1989). Petri Nets: properties, analysis and applications. Proceedings of the IEEE, 77(4), 541–580.

Peterson, J.L. (1981). Petri Net theory and the modelling of systems. Englewood Cliff, NJ: Prentice Hall.

Pineau, J., & Gordon, G. (2005). POMDP planning for robust robot control. In Robotics research, Springer Tracts in Advanced Robotics, Vol. 28, pp. 69–82.

Pito, R. (1996). A sensor based solution to the next best view problem. In Proceedings of International Conference on Pattern Recognition (ICPR), pp. 941–945.

Stachniss, C., Grisetti, G., & Burgard, W. (2005). Information gain-based exploration using rao-blackwellized particle filters. In Proceedings of Robotics: Science and Systems (RSS), Cambridge, MA.

Wang, J., Lewis, M., & Gennari, J. (2003). Interactive simulation of the nist usar arenas. In Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, pp. 1350–1354.

Yamauchi, B. (1997). A frontier based approach for autonomous exploration. In IEEE International Symposium on Computational Intelligence in Robotics and Automation .

Ziparo, V.A., & Iocchi, L. (2006). Petri Net Plans. In Proceedings of Fourth International Workshop on Modelling of Objects, Components, and Agents (MOCA), pp. 267–290, Turku, Finland. Bericht 272, FBI-HH-B-272/06.