Autonomous exploration for search and rescue robots

D. Calisi, A. Farinelli, L. Iocchi & D. Nardi
Dipartimento di Informatica e Sistemistica Università di Roma, "La Sapienza", Rome, Italy

Abstract

Search and rescue scenarios offer a wide variety of tasks where the experimentation of mobile robots is possible. Specifically, we describe some of the issues arising from scenarios where full autonomy of the robot is required. In this paper after briefly describing our robot configuration and basic functionalities, we address the exploration problem for a rescue robot and present a fully autonomous strategy for this task. We explain algorithms and methods used for this process and present typical experimental results.

1 Introduction

Robotic technology for search and rescue missions is a large section of robotics and artificial intelligence research. This type of applications involves different research areas, from mechanical design and sensors interpretation to perception, decision making, mapping, path-planning and victim detection.

Rescue robots are designed to strictly cooperate with human rescue operators assisting them during their missions. However, the reliability of the communication between the robot and the rescue operator represents a crucial factor in rescue missions. In many situations the robot should present some degree of autonomy, in order to effectively act in the environment also in those situations in which communication with human operators is either difficult or impossible. Moreover, autonomous capabilities such as safe navigation, map building and victim detection, can consistently help the human operators to control the robot.

Therefore, the motivation in providing a rescue robot with autonomous capabilities stems from the possibility of realizing a more effective system for use in rescue scenarios.

Our goal is to provide a rescue robot with basic autonomous capabilities in mapping, exploration, and victim detection, aiming at realizing a simple but very

WIT Transactions on The Built Environment, Vol 94, © 2007 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/SAFE070301

effective interface with an operator that can easily control a robotic platform by specifying only high-level commands, and can evaluate the status of the environment by analyzing information at a high level of description. Moreover, the system is robust to network failures and able to continue its mission even in the lack of communication with human operators.

To this end, we have implemented several basic capabilities that the rescue robot should perform autonomously: exploration, mapping, victim detection and localization. In the exploration process, autonomy is needed in order to increase reliability and effectiveness of such a task: indeed, many experiments have shown that an autonomous robot can perform a navigation task more quickly and safer than one controlled by a human, when the same sensor data are available. For the mapping process, autonomy helps the operator in interpreting and integrating the information coming from the on-board sensing devices, that would be very difficult due to the huge amount of such elementary data. Automatically generated maps are generated more quickly and are more precise than those that could be created by humans starting from the same input data. Victim detection and localization in the map is a task where autonomy is much more difficult to achieve given the difficulties arising in the rescue scenario. However, some forms of autonomy are useful in situations where direct communication with the robot is not possible and to reduce direct human control of the robot.

In this contribution we provide a brief overview of our mobile platform for autnomous exploration in search and rescue missions. In particular, we focus on the strategy for autonomous exploration and present some experiments validating our approach on our robotic platform.

2 Robot configuration and system architecture

The robotic platform we use for exploration experiments in rescue environments is a Pioneer ATX robot (shown in Figure 4), with four driving wheels and able to move over small obstacles. It is equipped with a SICK laser range finder, frontal and rear sonar rings, a stereo vision system, an infrared thermo-sensor and a voice transmission system. A common Pentium M laptop is used for on board computation. The robots has been tested within a Rescue arena built at the ISA laboratory in Rome and the implemented system was used during the RoboCup Real Rescue competition since 2004.

For programming complex behaviors on this robot, we use a highly modular software architecture [1] that allows for an effective and efficient integration of different modules and for easy reusing and team development. The modular architecture also allows for easily interchange modules as well as connected devices, ranging from actual robots to simulators. Modules are loosely connected to each other and can be scheduled independently with different priorities; interaction and communication among them occur using a centralized blackboard-type data repository. All modules publish their parameters and internal state for both supervision and debugging. Shared data can be exported remotely to the operator console and to team mates via a UDP or TCP link (802.11a wireless communication).

Besides the exploration and navigation module that are described in the next section, other important functionalities are implemented in the rescue robot.

A SLAM module [2] is used for building a consistent map of the environment during robot exploration. This module integrates information coming from the laser range finder, sonar sensors, and odometry. Victim detection module is used to detect victims and to report their location. This process integrates information coming from different sources: a stereoscopic camera, a thermo sensor, a voice transmission system used to receive ambient sounds.

Finally, the system provides a remote console to the rescue operator with a consistent interface for supervision and remote operation of the robot, and which publish several kinds of information: a metric map of the environment, the trajectory that the robot has followed during the mission, snapshots and other sensor data available for each detected victim and their location in the map. At the end of the mission all these information are collected and used to generate a detailed report of the mission, with useful data for rescuers.

3 Exploration and navigation

The exploration task is concerned with the navigation of an unknown environment, with the aim of collecting information about people, objects, or facts that are of interest for the rescue mission. In particular we concentrate on finding victims (i.e. human bodies).

3.1 The high-level exploration strategy

The exploration strategy needs to be very flexible, since the nature of rescue missions can be very different depending on the scenario at hand. Our solution has a hierarchical structure. At a higher level, a complex plan is used to determine the main behaviour of the robot. This plan is built using the Petri Net Plans formalism (see [3] for details), that makes it possible to define qualitative strategic rules to be applied in the mission; for example, if during navigation the robot detects a new victim, it stops navigation and goes toward the victim to determine his/her status. A graphical tool and a high-level automatic verification tool allow for easily define behaviors responding to the need of the mission at hand. In Figure 1 a particular instance of the exploration plan is depicted.

The exploration strategy can be divided into two main parts (as in a typical "next best view" algorithm [4, 5]): *i*) decide where to go next, considering that the environment has to be explored as fast as possible and that there are places in which there are more chances to find victims; *ii*) move the robot to the target position, that requires the ability to deal with cluttered and rough-terrains.

The first decision involves various issues, depending on the kind of environment to be explored. In a rescue mission, a robot has indeed different concurrent goals. In particular, one goal is to explore and build a consistent map of the environment, another one is to investigate further those areas, already been mapped, where there is the possibility to find victims. In our system, for what concerns the first goal, the choice of which position to explore is based on unexplored frontiers [6]. This method focuses on unexplored areas of the map by taking into account the unexplored frontiers (the bounds between unknown and free space), i.e. the positions that can be seen as "gates" to those areas. Moreover, in order to compute

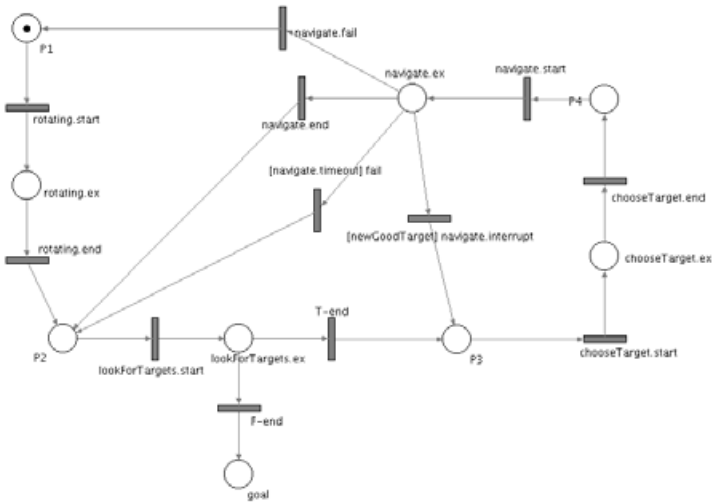


Figure 1: One instance of the exploration plan.

unexplored frontiers, we need only the current map. Another input for the choice of the next target to explore are the "interesting" areas, where it is possible to find victims, i.e. where it is needed a further investigation (e.g. the use of slower algorithms to process data). The problem can be seen as a *multi-objective search problem*, as typical of many robotic tasks [7]. Thus, we can use standard techniques based on information gain and make them easy to configure in order to take into account different importance weights for the different features to be measured in the environment. A detailed description of the method can be found in [8].

3.2 The navigation subsystem

The solution of the general motion planning problem is computationally very expensive, even in the most basic formulations. The so-called "generalized mover's problem" has been proved to be PSPACE-hard [9]. This motivates the development of heuristic algorithms that are able to quickly find a solution in many cases of interest, though often relaxing some requirements such as completeness and optimality.

Moreover, in a rescue environment, some of the assumptions, made in order to solve this problem, do not hold. For example:

- one cannot assume that the robot moves at a safe distance from the obstacles and so the real shape and size of the robot have to be considered in order to negotiate narrow passages;
- the sensors are not accurate and suffer from discretization errors; moreover, some maneuvers could unexpectedly fail, due to the presence of undetected obstacles, so one cannot assume to be able to do any kind of maneuver;
- the map is not known a priori.

A straightforward application of path-planning techniques is therefore not successful in a rescue domain. Most notably, the major difficulties are caused by the need of a very accurate maneuvering, but at the same time it is necessary to plan long paths, trying to avoid potentially difficult/blocked passages.

Our approach to navigation makes use of: *i*) a *global planner*, that takes into consideration the whole knowledge of the environment, and *ii*) a *local planner*, that takes as input only the sensor readings and a local small map, and can be more precise in steering the robot. In this way, we can decouple the problem by first solving a simple path-planning problem and then find a trajectory that follows the computed path; this is a widely used technique (see [10]).

The main novelties of our approach are the following: the global path-planner is a new combination of Probabilistic Roadmap and Growing Neural Gas, featuring the ability to be used with a map that is built incrementally while exploring; the local motion planner uses a Randomized Kinodynamic Planning approach which has been extended with interleaved planning and execution, feedback control and on-line pruning.

In the rest of the section we first introduce the local motion planner and then the global high-level path-planner. Details of this navigation subsystem can be found in [11].

3.2.1 The global path-planner

Using complete algorithms to find the topology of the environment (e.g. Voronoi diagram) is very expensive and, since we have a different map each cycle, a probabilistic approach is more convenient also for the global path-planner.

The most widely used probabilistic algorithm that builds a graph representing a roadmap of the environment is the Probabilistic Roadmap (PRM) [12]. The algorithm works by picking random positions in the configuration space and trying to connect them with a fast local planner. The problem with using this algorithm in a rescue environment is that it expects as input a map that does not change over time.

In order to overcome this limitation, we combine the PRM algorithm with Growing Neural Gas (GNG) [13]. GNG is a neural network with unsupervised learning, used to reduce the dimensionality of the input space. In this kind of network, usually nodes represent symbols and edges represent semantic connections between them; in order to make it fit our needs, we give them the same meaning they have in the PRM algorithm, i.e. we consider nodes as "places" and edges as the possibility to travel directly from one place to another. For the latter we use a simple straight-line local planner, usually used in PRM algorithms (i.e. if there is an edge between two nodes, it exists a straight-line path between them).

Our algorithm, that we call Dynamic Probabilistic Topological Map (DPTM), successfully combines PRM and GNG by taking into account the characteristics of the rescue environment. Moreover, thanks to GNG characteristics, the resulting roadmap contains very few nodes, with respect to a generic PRM on the same environment, thus making it easy to move and remove them as topology changes and to check if some edges are no longer valid, i.e. two nodes can be no longer connected using the straight-line local planner, and remove them.

3.2.2 The local motion planner

Our local motion planner is based on the Randomized Kinodynamic Planner (RKP) [14], which, in turn, is an extension of the well-known Rapid-exploring Random Tree (RRT) [15], that considers kinematic and dynamic constraints. These algorithms are probabilistic and build a tree, which quickly and uniformly explores the search space.

The search space of our approach is the set of poses (i.e. position and orientation) and velocities (speed and jog). We do not take into account dynamic constraints, due to the low velocities involved (caused by the fact that the environment is cluttered and partially unknown).

The use of an RRT-like approach allows for an easy specification of the set of constraints needed in order to navigate in the environment. Each constraint, indeed, needs only to be checked over the set of motion commands generated at each iteration. We can also specify which maneuvers are forbidden in a particular area of the environment (because, for example, they result in a stall, i.e. the robot is blocked by an undetected obstacle).

In general, the RRT and the RKP algorithms assume to have the whole map and that it does not change over time; consequently, they do not deal with the possibility to correct the plan once it has been computed and is being executed. In our case we have to deal with the imprecision and the discretization errors of the sensors and with the fact that the map is partially unknown; moreover, motion commands do not always lead to the desired behaviour, because of the roughness of the terrain, and because we cannot assume to be able to execute a motion command for exactly the amount of time needed. It is therefore necessary to consider a fast method to re-plan, if the current plan becomes invalid (i.e. it will bring the robot to a collision).

To overcome such difficulties we interleave planning and plan execution. This has some advantages over the two-phase approach, as the plan is generated when (and where) we have information about the environment and we can build the plan while the robot is moving; the results are very small plans, that are indeed just parts of the global plan.

In this way, we can use the algorithm to quickly move in the environment, without having to plan long trajectories, that in general will be made invalid during the subsequent exploration.

Though the low speed used to explore the environment causes only a small error in trajectory following and the use of a good localization and mapping method minimizes the input errors, a closed-loop control is still needed to correct errors in the trajectory execution. For this reason, we correct the current motion command using trajectory feedback. Another approach to use feedback in an RKP can be found in [16], where a single motion command (a feedback control law) is used at each iteration, thus making the method not suitable to accommodate constraints on the plan.

In narrow passages, the trajectory found on the tree becomes frequently invalid, due to control and sensors errors, thus a re-planning phase is needed. Anyway, if we maintain the tree, we can save a lot of computation. In [17] a different method is used, based on the computation of way-points along the trajectory. Since our environment is not as dynamic as the one considered in that paper, we can keep a lot more information (i.e. the whole tree) and "prune" only branches that contain

a collision. Since the robot is moving, we can also prune the root and all branches that do not belong to the current tree of possibilities. In this way we have, at each cycle, only a limited set of nodes and branches, from which we can continue to grow the tree, in order to further explore the search space. The trajectory is computed on the current tree and oscillations are avoided because at each node we only make a choice on which branch to follow; all other choices (branches) will then be pruned, thus limiting the size of the tree.

Since it is not always needed to plan complex maneuvers, for example in open spaces and flat terrain, we couple the RKP-based motion planner with a simple and quick reactive navigation algorithm, that takes into account only the current laser reading and does not do any planning, limiting its effect to the current motion commands. Thus, when planning is not required, the system is not overloaded with unnecessary computation and the robot can also increase its speed.

4 Experiments

In this section we present the description of a typical exploration task performed by our robotic platform. The exploration is totally autonomous, the platform builds a representation of the surrounding environment using our SLAM approach and decides which is the best location to go, to explore the whole environment following the approach described in the previous section.

The experimental scenario is an indoor unstructured environment, that follows the NIST (National Institute of Standards and Technology (www.nist.gov) standard for robotic performance evaluation.

Pictures in Figure 2 show the information that the robot acquire and send to the base station during the mission execution. In particular, the map is composed by three colour, black represents obstacles, white represents free space and blue (gray) represents unexplored space. The square represents the postition of the robot estimated by the SLAM algorithm. The cross represents the next target point for the robot and the line connecting the robot to the cross represents the planned path. Finally, the colored (light gray) points represents the current frontiers that the robot extracted from the map.

As it is possible to see, the robot correctly tries to reach the nearest available frontier. As soon as a frontier becomes totally explored the robot is able to plan a new target point, without having to reach the previous location. In this way the exploration can quickly proceed towards unexplored regions of the environment.

Last picture in Figure 2 shows the final map and the actual path that the robot followed. The map has been completely explored with a very effective strategy.

As for the navigation, Figure 3 shows a narrow passage, that the local motion planner is able to deal with. In the first two images the trajectory found by the motion planner causes the robot first to move backwards, because it cannot turn or go forward; then, it find its way in the narrow passage. In the third and fourth image, while the trajectory is being followed, the robot is shifted to the left (e.g. the localization module has corrected the robot position on the map), thus making the trajectory invalid and requiring the computation of a new one. The whole process is done in parallel with the robot motion, i.e. it is never stopped for re-planning (unless the collision is found in the current step). The robot navigates at a speed

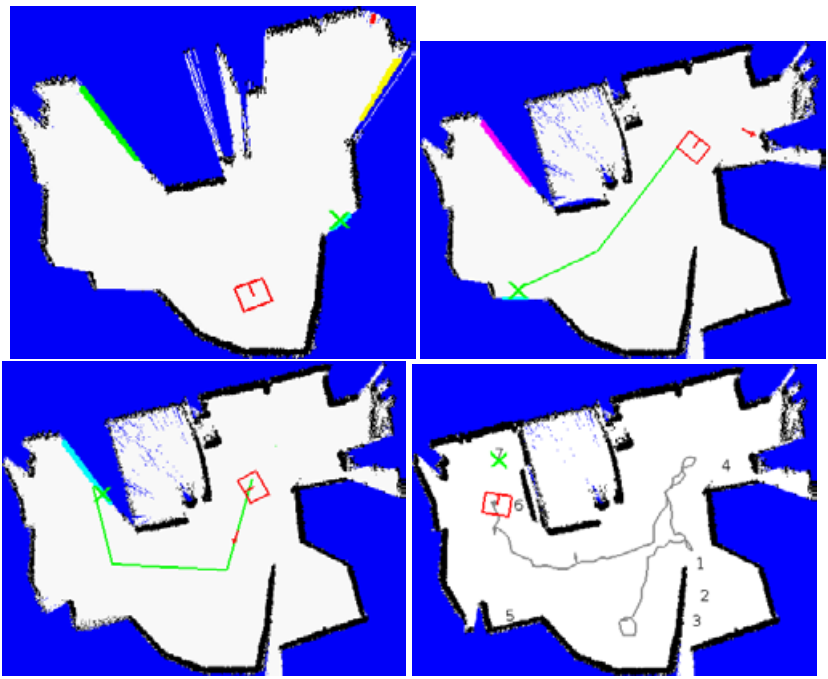
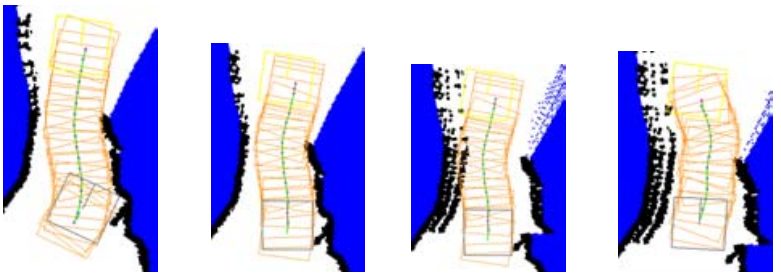


Figure 2: An example of an autonomous exploration mission.



(a) 1 (b) 2 (c) 3 (d) 4

Figure 3: The local motion planning dealing with a narrow passage.

of 10 cm/s and its size is 48x50 cm, while the narrow passage is only 60 cm wide (the map is discretized at 50 pixels per meter).

Finally, Figure 4 shows the map of a yellow arena built in our laboratory, which has been explored fully autonomously by our rescue robot. The robot start its mission in the places marked with a thick arrow, in the upper right corner of the image. You can see the path done by the robot during the exploration and the two victims successfully detected.

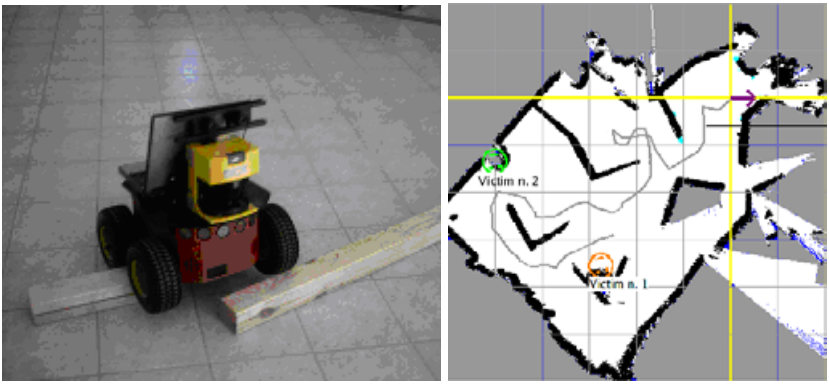


Figure 4: The robot and the map of the arena in our laboratory.

5 Conclusions and future works

The robotic system described in this paper allows for a fully autonomous exploration and mapping task. Autonomy is important not only for its robustness to communication failures, but also for speeding up the whole process of victim searching and to provide user operator with a more flexible and effective platforms.

Exploration strategy can be improved by including high-level semantic information about the environment. In this context, we are working to include Victim Detection-oriented decision in the choice of the next target position to explore, in order to increase the chances of find victims, i.e. explore first those areas in which the probability of finding victim is higher.

Moreover, we are investigating the opportunity to use multiple heterogeneous robots for the exploration task. In this case the main issue to deal with is coordination among the robotic platforms in order to optimize resource allocation and minimize conflicts during the exploration mission.

Finally, additional extensions in other modules are also desired. For example, we are working on extending the mapping process in order to generate 3D visually realistic maps. This will increase the applicability of the proposed technology in search and rescue scenarios.

The presented approach is a first step towards building consistent maps and detect victims located in these maps. A more systematic analysis of the performance of the mapping and the victim detection processes are important to evaluate the entire approach to robotic search and rescue missions.

References

[1] Farinelli, A., Grisetti, G. & Iocchi, L., Design and implementation of modular software for programming mobile robots. *International Journal of Advanced Robotic Systems*, **3(1)**, pp. 37–42, 2006. ISSN 1729-8806.

[2] Grisetti, G., Stachniss, C. & Burgard, W., Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.
[3] Ziparo, V.A. & Iocchi, L., Petri net plans. *Proc. of Fourth International Workshop on Modelling of Objects, Components, and Agents (MOCA)*, Turku, Finland, pp. 267–290, 2006. Bericht 272, FBI-HH-B-272/06.
[4] Pito, R., A sensor based solution to the next best view problem. *Proc. of Int. Conf. Pattern Recognition, (ICPR)*, pp. 941–945, 1996.
[5] González-Baños, H.H. & Latombe, J.C., Navigation strategies for exploring indoor environments. *I J Robotic Res*, **21(10-11)**, pp. 829–848, 2002.
[6] Yamauchi, B., A frontier based approach for autonomous exploration. *IEEE Int. Symposium on Computational Intelligence in Robotics and Automation*, 1997.
[7] *Proceedings of IEEE/RSJ IROS 2006 Workshop on Multi-objective Robotics (IROS-MOR 2006)*, 2006.
[8] Calisi, D., Farinelli, A., Iocchi, L., Nardi, D. & Pucci, F., Multi-objective autonomous exploration in a rescue environment. *Proc. of IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*, Gaithersburg, MD, USA, 2006.
[9] Reif, J.H., Complexity of the mover's problem and generalization. *Proc. 20th IEEE Symp. on Foundations of Computer Sciences (FOCS)*, pp. 421–427, 1979.
[10] Latombe, J.C., *Robot Motion Planning*. Kluwer Academic Publisher, 1991.
[11] Calisi, D., Farinelli, A., Iocchi, L. & Nardi, D., Autonomous navigation and exploration in a rescue environment. *Proc. of the 2nd European Conference on Mobile Robotics (ECMR)*, Edizioni Simple s.r.l., Macerata, Italy, pp. 110–115, 2005. ISBN: 88-89177-187.
[12] Kavraki, L. & Latombe, J., Probabilistic roadmaps for robot path planning. *Practical Motion Planning in Robotics: Current Approaches and Future Challenges*, K.G. and A.P. del Pobil, pp. 33–53, 1998.
[13] Fritzke, B., A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems 7*, eds. G. Tesauro, D.S. Touretzky & T.K. Leen, MIT Press: Cambridge MA, pp. 625–632, 1995.
[14] LaValle, S. & Kuffner, J., Randomized kinodynamic planning. *Proc. IEEE International Conf. on Robotics and Automation*, pp. 473–479, 1999.
[15] Kuffner, J. & LaValle, S., Rrt-connect: An efficient approach to single-query path planning. *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000), San Francisco, CA, April 2000.*, 2000.
[16] E. Frazzoli, E.F., M.A. Dahleh, Real-time motion planning for agile autonomous vehicles. *2000 AIAA Conf. on Guidance, Navigation and Control.*, 2000.
[17] Bruce, J. & Veloso, M., Real-time randomized path planning for robot navigation. *Proceedings of IROS-2002, Switzerland, October 2002*, 2002.