Autonomous Systems for Search and Rescue

Shahram Bahadori, Daniele Calisi, Andrea Censi, Alessandro Farinelli, Giorgio Grisetti, Luca Iocchi, Daniele Nardi, Gian Diego Tipaldi

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
E-mail: `<lastname>@dis.uniroma1.it`

Abstract. Search and Rescue scenarios offer a wide variety of tasks where the experimentation of robots is possible. In particular, we address 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 the experimental scenario, we address three main components of an autonomous rescue robot: mapping and localization, navigation and exploration, and human body detection. Specifically, we explain algorithms and methods used by our modules in order to achieve a significant degree of autonomy in mapping and exploration of unknown rescue scenarios and in victim detection therein.

1 Introduction

The deployment of robotic technology in search and rescue missions for assisting the rescuers covers a large section of robotics and artificial intelligence research. The design of systems that are suitable for this type of applications involves different research areas, from mechanical design and sensors interpretation to perception, decision making, mapping, path-planning and victim detection.

Although rescue robots are designed to strictly cooperate with human rescuers during their missions, the reliability of the communication between the robot and the rescue operator represents a crucial factor. 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. Therefore, the motivation in providing a rescue robot with autonomous capabilities stems from the possibility of realizing a more reliable and effective system for use in rescue scenarios.

Some basic activities that the rescue robot should perform autonomously are: mapping, navigation, victim detection and localization. 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 to understand due to the huge amount of such data. Automatically generated maps are built more quickly and are more precise than those that could be created by humans starting from the same input data. As for navigation, many experiments have shown that an autonomous robot can perform a navigation task more effectively and safely than one controlled by a human, when the same sensor data are available. Finally, increasing the autonomy of a rescue robot will help the operators to easy control multiple robots with a high-level control interface.

Other research groups involved in the development of rescue robots have realized some autonomous capabilities in their robots. For example Alcor [38] has autonomous navigation and mapping capabilities, while the exploration strategy is guided by a saliency map and is represented via a high-level program written using the Golog language. RFC Uppsala [46] is developing a fully autonomous multi-robot system, including victim identification based on heat and CO_2. IUB [3] has an autonomous mapping module based on a laser range finder and an autonomous victim identification module based on an infrared camera and a shape recognition algorithm. CASualty [23] has one fully autonomous robot and a teleoperated one, though it present modules for autonomous localization, mapping and victim identification as an aid to the operator. Deutschland1 [31] has an autonomous 3D mapping algorithm based on a tilted laser scanner and an ICP based method. The method requires the robot to stop the navigation in order to take the 3D scan. RescueRobots Freiburg [26] has a SLAM method based on Scanmatching and RFIDs actively distributed in the environment by the robot itself. This group has also developed a visual odometry module that act as a base for the Scanmatching module as well as a behaviour-based autonomous navigation algorithm. Thanks to a tilted laser range finder, they are able to build 3D maps of the environment, labelling semantic places in order to navigate in non-planar environments. Victim identification in all teams usually makes use of an infrared camera and methods like shape recognition, skin detection or motion detection. Autonomous mapping is performed using sensors in the class of laser range finders.

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 effective interface with an operator that can control a team of rescue robots by specifying only high-level commands and can evaluate the status of the environment through a high-level, user oriented description. Moreover, the system will be robust to network failures and able to continue its mission even in the lack of communication with human operators.

In this article we first describe the functionalities of the system, the software architecture, and the robot configuration and experimental environment on which we have tested our methods. Then we provide a description of the methods developed for navigating and exploring the environment, for building accurate and consistent maps and for detecting victims.

2 System functionalities

In order to have an autonomous robot system the functionalities that we address are as follows:

SLAM and Sensor fusion. We solve the SLAM problem using only the laser range finder sensor. This estimates the trajectory followed by the robot; a sensor fusion module integrates data coming from the laser range finder and sonars, producing occupancy grids for autonomous navigation and human operator.

Navigation. The navigation module uses information provided by the SLAM module and moves the robot to goal positions reporting possible failures. It has to deal with dynamic obstacles, map incompleteness, movements that unexpectedly fail, and so on.

Exploration. The exploration module plans a sequence of goal positions ensuring successful coverage of the area to be explored.

Victim detection. This module identifies victims and report their location based on the data from: 1) a stereoscopic camera; 2) a thermo sensor; 3) a voice transmission system used to detect sounds from the environment.

Human-Robot Interface. 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.

The desired behaviour of the robot is specified as follows: when communication is working, information processed from the sensors are transmitted continuously to the remote console and the operator can evaluate them in real time and interact with the rescue robot through high level control commands; in case of communication breakdown, the robot will continue its tasks in an autonomous way, by exploring the rest of the map, providing information about possible victims and finding its way back to the starting position once all the map has been explored.

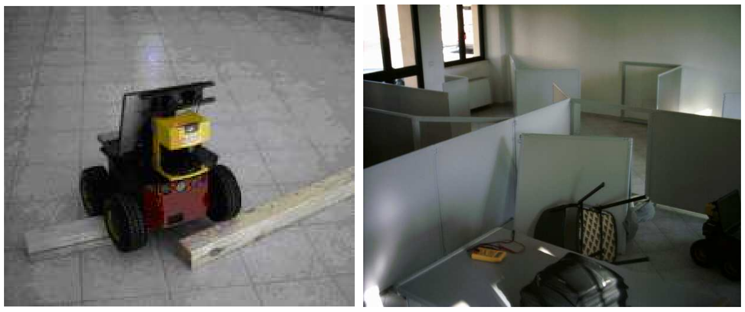

Fig. 1. Rescue robot and arena used for experiments.

2.1 Robot configuration and experimental setting

Experimental environment. Experiments have been performed within a Rescue arena built at the ISA laboratory in Rome and the implemented system was used during the RoboCup 2004-2006 Real Rescue competition. We also use a simulated environment built using the USARSim [40, 2] framework, where we modeled the real mobile robot and sensors. The use of a simulated environment, very closely representing the real one, provides for a quicker development and tuning of the algorithms and parameters, as well as for the possibility to run batch tests (e.g. for learning) without having to worry about usual real robot and devices problems.

Robot configuration. The Pioneer3-AT robot (Figure 1) has four driving wheels and it is 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.

Software architecture. We use a highly modular software architecture [14] that allows for an effective and efficient integration of different modules and for easy reuse 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 supervision, parameter tuning and debugging. Shared data can be exported remotely to the operator console and to team mates via a TCP or a UDP link (802.11a/g wireless communication).

3 SLAM and sensor fusion

The Simultaneous Localization and Mapping (SLAM) problem has been deeply investigated by the robotics community in the last decade [18, 6, 20, 10, 32, 47, 12, 34]. In the last years, several authors started to consider full 3D representation, relaxing the planar assumption and dealing with more complex environments and rough terrain [41, 35, 36]. It is considered a complex problem, because for localization a robot needs a consistent map and for building a map it needs a good estimate of its location. This mutual dependency among the pose and the map estimates requires to search for a solution in a high-dimensional space. Although several effective techniques have been proposed so far, a general approach that works with any sensor setting and in arbitrarily unstructured environments is still under investigation.

In the context of the RoboCup Rescue competition several teams approached the mapping problem [37, 9, 4, 25]. In particular, in [37] the use of a 3D laser range finder is described, while in [9] the occupancy grid was generated using sonar or infrared proximity sensors. In [4] two mapping algorithm are fused: an occupancy grid framework relying on odometry data and an RBPF mapper [18]. [25] uses visual odometry and scan matching from a laser range finder.

In the following, we discuss the many challenges of the rescue environment and how we addressed them. Subsequently, we present our approach for mapping planar and semi planar rescue environments, by integrating the output of different sensors. Finally, we discuss some of the results obtained by using our approach, performing a comparison with state-of-the-art mapping techniques.

3.1 Challenges of rescue environments

Rescue environments present a set of challenges that are not considered in most of the SLAM approaches proposed in literature.

Unstructured Environments: first, the amount of structure present in rescue environments is supposed to be significantly lower than the one present in environments for which most of the existing approaches are developed and tested. The lack of a structure makes it hard to use a prior knowledge about the environment and can cause the SLAM process to fail.

Non Planarity: most of the SLAM approaches assume to operate in a planar working space. The problem of considering full 3D scenarios must be approached not only with specific sensors and techniques, but also with an adequate mechanical structure.

Material heterogeneity: rescue environments are built with different materials, and it is difficult to choose a sensor able to detect all of them, with a precision suitable for building an on-line map. This imposes to integrate the output of heterogeneous sensors to compensate the weaknesses of a single device.

Environment Size: a key problem in mapping is to provide consistent maps, i.e. maps in which sensor information in different time frames are not conflicting. This is particularly evident when a robot revisits a known place after navigating for a long time in unknown terrain. The probability of facing this problem depends directly on the size of the explored environment.

Human comprehension: finally, most of the SLAM approaches focus on having maps that are accurate enough for being interpreted by a program, rather than by a human. The human operator prefers to have dense maps rather than feature maps, and usually wants to have a real time feedback of the mapping process.

With respect to the above issues, our choice in the design of our SLAM method has been to consider unstructured environments with limited non-planarity and made by heterogeneous materials. In particular, our approach generates easy human-readable maps using a scan matching approach. Since in the experimental settings we only consider limited size for the environment, using an accurate sensor like the LRF significantly bounds the error. This results in limited inconsistencies, which do not compromise the readability of the final map. The quality of the obtained maps justifies the use of a simple and efficient SLAM technique that does not consider the loop closure problem.

3.2 Mapping planar environments

As many works on SLAM have shown [20, 21, 43, 33, 32], a good setting for building maps in indoor environments is to use a wheeled robot equipped with a LRF. The high accuracy of this kind of sensors allows to build accurate planar maps.

Most of the works on SLAM deal with the loop closure problem, since only when revisiting known locations the error accumulated when mapping new areas can be reduced. When the environment is large, we can use other approaches like the ones described in [18], which has been proved to work properly if joined with an active exploration technique [45]. Although the use of a standard SLAM technique works in a rescue setting, the computational burden derived from the use of such a technique does not allow the concurrent execution of other tasks on the PC controlling the mobile robot. In [19], two efficient optimizations for Rao-Blackwellized SLAM have been developed, allowing real-time mapping of large environments.

From the analysis of the common techniques for SLAM, we devised and implemented a mapping method that shows good performance in rescue environments. The method is an incremental scan matching technique that aligns

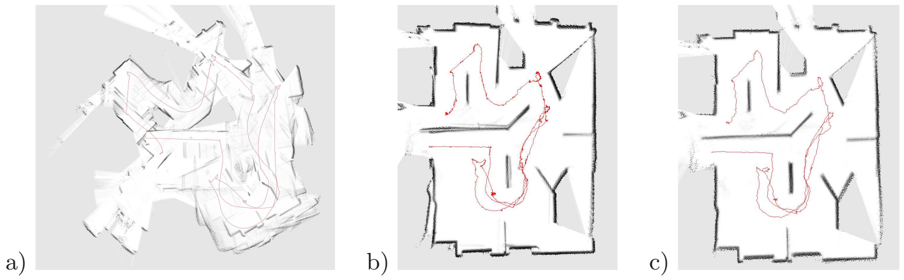
Fig. 2. Occupancy grid map generated from data acquired in a yellow rescue arena. Results obtained using the position estimate computed by a) using the raw odometry data b) performing scan match c) using a Rao-Blackwellized mapper.

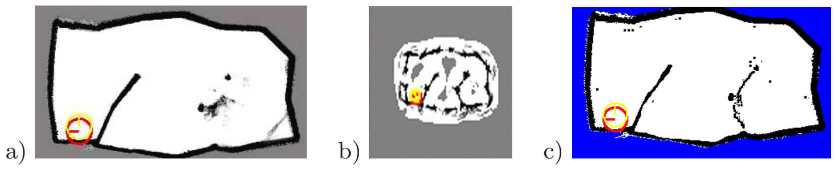
Fig. 3. Maps obtained by different sensors in a rescue environment: a) laser map b) sonar map c) integrated map. The glass objects are invisible in the laser map while they are visible in the sonar map. The robot uses the final integrated map in c) for navigation.

each scan with the previously accumulated ones, by performing a local search around the odometry estimated position, trying to maximize the past and current scan overlapping. As a difference from the Lu and Miles approach [32], our scan matcher does not rely on the correspondence among the single readings. Scan matching can be seen as solving the following maximization problem

$$\hat{p}_t = \underset{p_t}{\operatorname{argmax}} \ \mathbf{score}(z_t, m_{t-1}, p_t). \tag{1}$$

Here the function **score** measures how well the current scan z_t fits in the previous step map estimate m_{t-1}, given the robot pose p_t.

In our approach, the initial guess is computed from the last odometry reading and the previous pose estimate $\hat{p}_{t-1}$. The score function used by our scan matching algorithm is the following:

$$\mathbf{score}(z_t, m_{t-1}, p_t) = -\sum_i^n \mathcal{M}_{t-1}(R_{\theta_t} z_t^i + T_t). \tag{2}$$

Here $z_t^1 \ldots z_t^n$ are the points that belong to the scan z_t, $\mathcal{M}_{t-1}$ is a function obtained by convolving a point map m_{t-1} with a Gaussian kernel, R_{θ_t} and $T_t = (x_t \ y_t)^T$ are the rotation matrix and the translation vector computed from pose hypotheses $p_t = (x_t \ y_t \ \theta_t)^T$ respectively. The maximization is performed each time a new scan arrives, using gradient descent. Once the new pose $\hat{p}_t$ is computed, the new map m_t is estimated by integrating the current scan z_t in the previous map m_{t-1}.

It is very similar to the approach used by the Carmen scan matcher (Vasco) [1], but it embodies some differences in the map representation. Instead of considering the map as a grid of cells whose centers are fixed, in our representation the center of a cell can be moved. In particular the center of a cell is computed as the mean of points falling in that cell. This allows our approach to reach a sub cell precision, thus improving the overall performance.

3.3 Sensor fusion

Despite to its precision a LRF cannot deal with any material; for example glasses and mirrors (typically present in a rescue environment) introduce large errors in the mapping process when only a LRF is used. For this reason a standard SLAM technique, which relies on raw laser data cannot be directly applied. We approached this problem by integrating the LRF with sonar sensors, as further described below.

In order to deal with the problem that LRF cannot detect mirror or glasses, we used ultrasonic sensors that, although less reliable and accurate than LRF, allow for dealing with these materials. The strategy we have adopted has been to discard all of the laser readings falling in the sensor cone, whose values can not be explained by a corresponding sonar reading.

Our integration process works as follow. The robot pose is evaluated only using the LRF data. Based on this pose estimate two different occupancy grids are constructed: one using the laser data and one using the sonar data. The integration process works incrementally, each time a sonar reading comes, the spanned cells in the laser map are analyzed. If an occupied region in the laser map is near to the sonar reading endpoints is found, the reading is "explained", and should not be considered for generating the final map, otherwise it is "unexplained". The final map is obtained by adding to the laser map the occupied regions of a sonar map obtained by considering only the "unexplained" readings. In this way, we are able to recover several situations in which the LRF cannot detect a glass element in the environment, as shown in Figure 3.

3.4 Results

The experiments have been performed by comparing our scan matching based method and a Rao-Blackwellized mapper [18, 45]. As expected the Rao-Blackwellized is more accurate when closing loops, while our scan matching method is more efficient. Moreover, the quality of the map provided by our approach method is good enough for a rescue robot.

Figure 2 provides an example of the maps generated by the implemented methods. The first image shows the odometry error of the robot in the environment; Figure 2b shows the results of our scan matching method; and Figure 2c shows the results of the Rao-Blackwellized algorithm. The range data are acquired through the combination of values coming from a LRF and a set of ultrasonic sensors. The figure also shows both that the Rao-Blackwellized algorithm is more accurate than scan matching and that this difference is acceptable for the task of returning a map of an environment by a rescue robot.

On the other hand, computational time for the scan matching method is significantly reduced. Typical cycle time for processing data on our robot is 10 ms for the scan matching method against 100 ms for Rao-Blackwellized algorithm.

4 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).

4.1 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 [50] 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 4 a particular instance of the exploration plan is depicted.

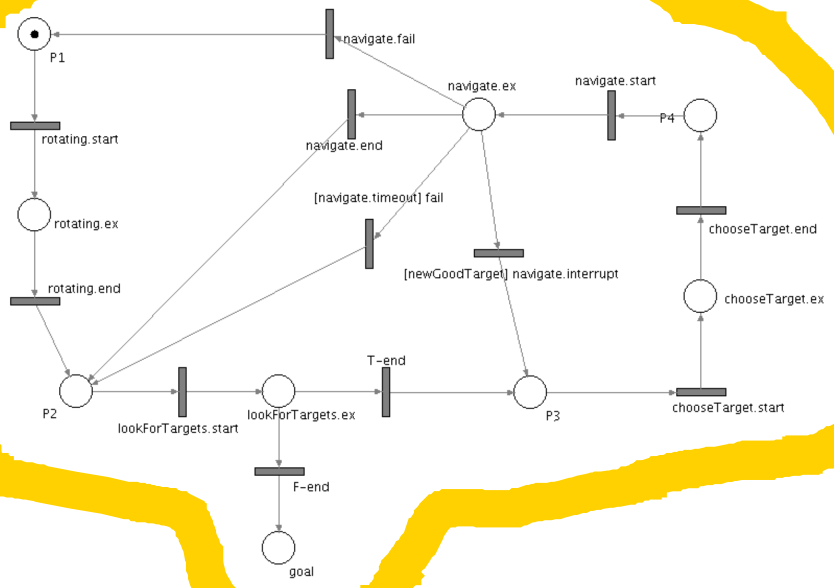

Fig. 4. One instance of the exploration plan

The exploration strategy can be divided into two main parts (as in a typical "next best view" algorithm [39, 17]): *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 [48]. 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 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 [22]. 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].

4.2 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 [42]. 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 manoeuvres could unexpectedly fail, due to the presence of undetected obstacles, so one cannot assume to be able to do any kind of manoeuvre;
– 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 [29]).

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 whih 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 [7].

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) [24]. 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) [15]. 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.

The Local Motion Planner Our local motion planner is based on the Randomized Kinodynamic Planner (RKP) [30], which, in turn, is an extension of the well-known Rapid-exploring Random Tree (RRT) [28], 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 manoeuvres 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 [11], 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 [5] 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 manoeuvres, 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.

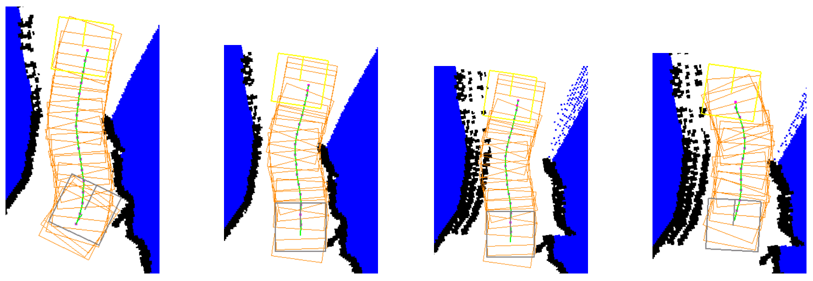
(a) 1 (b) 2 (c) 3 (d) 4

Fig. 5. An example

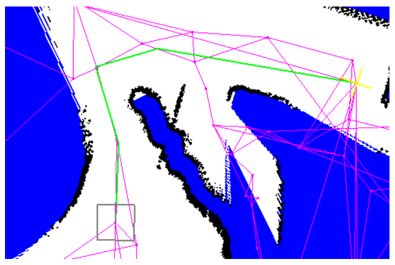

Fig. 6. The DPTM

4.3 Results

In Figure 6 a partial map is shown and the correspondent DPTM built on it. The thick line is the topological path, computed on the DPTM, that has to be followed by the motion planner. In Figure 5 we can see an example showing 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 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).

5 Human body detection

In this section we provide an overview of the method for Human Body Detection we have used on top of our rescue robot. The overall process is sketched in Figure 7. The method takes as input a stereo image. The left image is processed with an edge detection module and then with contour extraction. These steps provide a segmentation of the image based on closed contours. Stereo pair is also processed by a stereo algorithm to compute the disparity [27] and 3D information about the segments identified before. Body part classifier takes these segments and uses a similarity measure to detect the body limbs, that are finally used for matching a human body model.

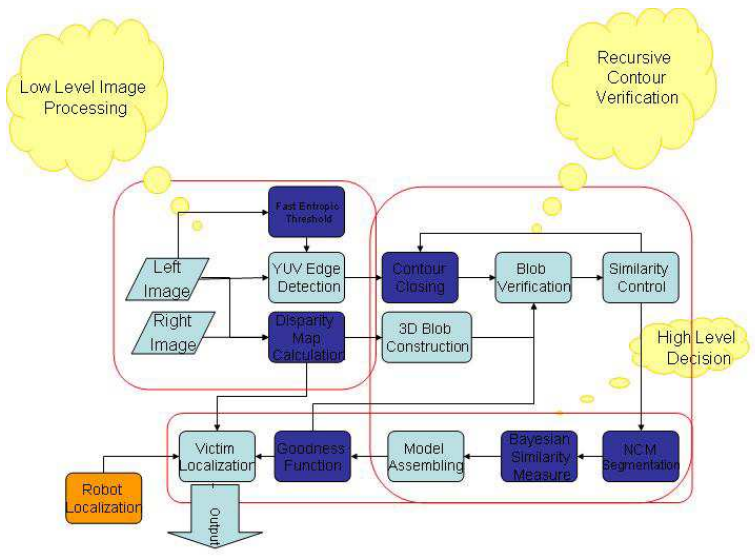

Fig. 7. Human Body Detection Process

5.1 Image Segmentation

The algorithm uses YUV color space with dynamic threshold [13] for the edge detection phase and a gap-filling process [16] to compute a set of closed contours. Moreover, 3D information are also used to filter out closed contours with large variation in disparities, representing non-compact 3D segments. Finally, for each closed contour, 3D measurements are taken for the subsequent model matching phase: i.e., length, width, and thickness.

The first steps of the process are shown with a sample image in Figure 8. The original image and its disparities are shown respectively in Figures 8a) and 8b), while the result of edge extraction and gap filling is in Figure 8c).

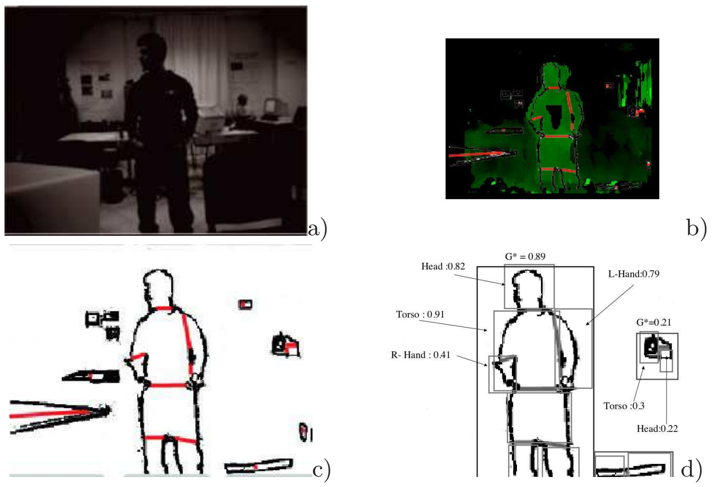

Fig. 8. Segmentation example.

5.2 Recursive contour verification

After segmentation, a recursive algorithm is used in order to achieve the best configuration for HBD. The main goal of this algorithm is to find the best configuration of the 3D segments in order to use them for the similarity measure. Mainly it should merge very small segments and split very large ones considering the position of the segments and the natural decomposition of the human body model [44]. The definitions of very small and very large are in [16]. The output is a set of closed 3D contours that are big enough to be considered as body limbs (Figure 8 c). After RCV, we apply the shortcut rule [44] to determine the natural decomposition of the human body (Figure 8c red lines).

5.3 Bayesian Similarity Measure

The next phase consists in measuring the similarity between the detected segments of body limbs and the parts of a 3D human body model. A cost function is defined to measure the similarity of body parts with the Bayesian probability that the segment belongs to one of the object classes present in the model. More specifically, we denote with $B = \{b_1, b_2, ..., b_m\}$ the set of the body limbs in the model also including relations among their dimensions. Each detected segment can have more than one matching with body limbs in the model.

The body part identification problem is to match a set of 3D segments extracted from images as explained before against a set of model body parts. The set of 3D segments detected in the image, denoted with $S = \{s_1, s_2, ..., s_k\}$ includes 3D measurements such as length, width and thickness of each segment s_j. Also the body limbs in the model b_i include 3D information about width, height, thickness that have been obtained from standard medical data

From B and S we compute a matrix $D = \{d_{11}, d_{12}, ..., d_{mk}\}$ containing limb similarities, where each $d_{ij} \in [0, 1]$ measures the similarity between the detected segment s_j and the body part in the model b_i. We then represent with $H =< D, V >$ the combination of the similarity matrix D and the information about the pose of the person $V \in \{vertical, horizontal\}$ in the image.

The best hypothesis H^* for limb detection is computed as (see [16] for details):

$$H^* = arg \max_H P(S|H,B)P(B|H) \tag{3}$$

The likelihood $P(S|H,B)$ is measured by considering the similarities in sizes and relative positions between the set of decomposed segments and the model body parts. Note that the model parameters are not fixed but are dynamically determined based on the matched hypotheses. This means that the matching procedure selects not only the best match pairs, but also the best human model level to describe the extracted 3D segment. For example, in the evaluation of an hypothesis with a high score for head and torso we use the model at middle level (Head, Upper and Lower) while if there is also a segment with high score as a leg we use the lower level of model (see Figure 9). The conditional probability $P(B|H)$ is estimated based on the number and types of matches between extracted segments and body limbs in the model. The presence of a fine level body part indicates a higher likelihood than that of a coarse level body part does. Although the head and the torso do not have subparts, their appearance imply a higher likelihood than the appearance of other body parts such as two arms.

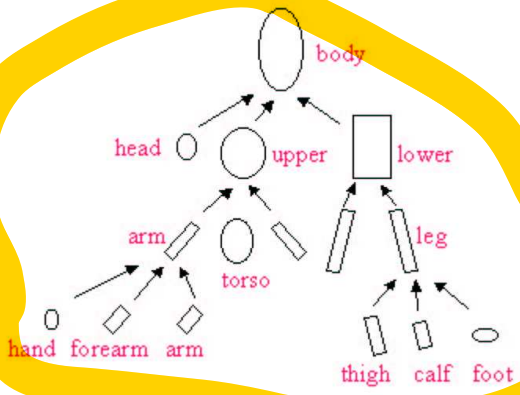

Fig. 9. Body hierarchical model

5.4 Model Matching

The model that we use is a hierarchical model (see Figure 9) that in the top considers the body as a blob that respects the relations of the length, width and height of the human body [16]. All of the limbs are calculated based on the relations of the measures and the positions of the joints.

Model assembling is performed by evaluating the quality of the best matching hypothesis H^* computed as described in the previous paragraph. To this end we define a goodness function $G_M(\cdot)$, that measures the quality of a matching hypothesis over the model M. This model is composed by two components: the first component is a table of limbs measure relations and the second one is a table with joint relations.

The definition of the goodness function in this step is important as it should be precise to measure the correctness of the assembled model, while it should also handle noise and model deformations. We use the Bayes rule to define the goodness function as an extension of the function proposed by Zhao [49]. The main difference is in the use of the probabilistic distribution model of body limbs instead of the entire body model in the final evaluation of the goodness function; it makes the model less sensitive to the missed body limbs and external occlusions. Furthermore, the function that we use takes into account the dimensions of body limbs and the position of body joints in a 3D space. We also added a noise factor to the model of the 3D distribution to decrease the noise effect on the model assembling. This noise factor comes from the segmentation and determines the level of sensitiveness of the goodness function.

If the goodness $G_M(H^*)$ is above a pre-defined threshold then the hypothesis is accepted and the presence of a human figure in the image is declared, otherwise a new iteration is attempted (see Figure 7). The process terminates with the result that a human figure is not present in the image when, after a limited number of iterations, none of the segments has a sufficient score for a human body limb, so the final score of the goodness function is low. The number of these iterations is due to the level of details that we need to consider. In our experiments we perform at most five iterations.

5.5 Results

To evaluate the developed system we have first collected an experimental data set containing several sequences of stereo images, that are classified in four groups: 1) image pairs without presence of people to test false positives; 2) image pairs with the presence of full bodies; 3) image pairs with partially occluded body limbs (but with visible heads or torsos); 4) image pairs with only body limbs (heads and torsos are not visible).

We performed two kinds of evaluation: *category evaluation* and *random evaluation.* Category evaluation has been performed by configuring the method's parameters to have the best results on a specific category of the bodies; for each category we first calibrate the system using a set of ten image pairs of the same category and them run the algorithm on the entire category. These experiments are necessary to achieve the best results by category and define the upper bound and limitations of the method. For the random evaluation we use parameter configuration that performed the best results in the category phase. According to our experiments, the parameters used for the partially covered set give the best performance as general setting for all the categories.

Table 1 shows the experimental results, indicating the percentage of correct classification of category and random evaluation from horizontal and vertical views of the bodies:

The results are slightly worse for random evaluation with respected of category evaluation, but still sufficiently good in most of the cases, except for the Full Covered data set. Since the algorithm uses the position of head or torso as reference point to determine the position of the joints, in case of Full Covered bodies it cannot determine them correctly, thus the method is not adequate in

	Category H	Category V	Random H	Random V
Full	99 %	99%	95 %	99 %
Partially Covered	95 %	95 %	95 %	95 %
Full Covered	48 %	82 %	38%	60 %
No Victim	99 %	99 %	97 %	97 %

Table 1. Correct classification rate: Category Results Vs Random Results for vertical and horizontal posed bodies

this cases. The results are better in the vertical position, because it is easier to find the position of the joints and the relation of the limbs in this case. The lower scores in the horizontal positions are due to the diagonal poses with auto occlusions; when the auto occlusion is high the 3D blobs and shortcut rule cannot define the natural decomposition of the body limbs.

The algorithm takes approximately 130 ms on an Athlon XP 2800+ with 512 MB of RAM to process each frame, varying from 85 ms to 180 ms depending on the number of objects to examine and on image noise.

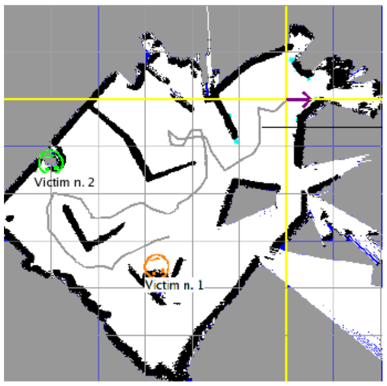

Fig. 10. The map of a Yellow Arena built in our laboratory, showing also victims locations and the path done by the robot while autonomously exploring it.

6 Conclusions and future works

Our system allows a fully autonomous mapping and localization and a semi-autonomous victim detection. Such features speed up the whole process of victims searching. Below we focus on the improvements of the system components, that we have specifically addressed in this paper: mapping and localization, exploration and navigation, and victim detection.

6.1 Mapping and localization

In order to improve mapping task, we are currently analyzing mapping methods for non-planar environments. When the robot that maps the area does not move on a plane, the problem of SLAM becomes more difficult, for the following reasons:

– The robot pose space grows from three to six dimensions. If we want to track the robot pose with a Rao-Blackwellized particle filter, we need a cubic number of samples to have the same coverage of the pose space.
– If we are using a wheeled robot, the odometry sensors provide a bad estimate, when moving on a non regular surface.
– The laser range finder, that can be safely used in planar environments looses its effectiveness, since it can only detect obstacles lying on the scanning plane; therefore, we must switch to more inaccurate sensing device like stereo cameras.

For the above reasons, a proper sensor equipment for a 3D mapping robot includes an Inertial Measurements Unit (IMU), for dead reckoning, and a stereo camera as eteroceptive device. Moreover, since the stereo camera provides sparse and noisy information, it is extremely difficult to apply some dense sensor matching technique (like scan match), and feature based SLAM techniques must be used. In order to provide the operator with a feasible environmental representation for operation, a reconstruction technique can be used on the path estimated by the SLAM algorithm.

The state transition is governed by the odometry and the IMU. At each time a local view of the landmarks is built from the estimated pose and the map, and the data association is solved using the nearest neighbour principle as in many SLAM approaches. To increase the robustness of the approach with respect to association failures, FastSLAM can be effectively used, provided that the sample space dimension has been reduced, by exploiting the information provided by the IMU.

We are currently performing a deeper analysis and experiments in order to devise a suitable configuration and the appropriate technique that can be effectively used in non-planar environments.

6.2 Exploration and Navigation

We presented an approach to autonomous exploration that is based on a high-level plan and the use of multi-objective techniques. Thanks to the use of the Petri Net Plans formalism, the high-level plan can be easily adapted and modified in order to fit the particular needs of the mission at hand. Moreover, also the weights coefficients, used to choose the next area to explore, can be modified to bias the search towards unexplored areas or places where victims are likely to be found.

The navigation subsystem is based on two probabilistic methods that have been suitably extended in order to adapt to a rescue environment. The resulting modules are able to steer a robot in a cluttered environment and have been tested both in simulation and in the real rescue arena. The use of a topological representation in the global level makes it simple to interact with an operator or with a cognitive level. Moreover, the use of an RRT-like algorithm make it straightforward to introduce constraints on robot moving capabilities, like those on minimum turning radius or on forbidden manoeuvres in particular areas of the arena (due to undetectable obstacles).

A very interesting and challenging future direction of work for exploration is to extend our technique to Multi-Robot Systems. The use of a group of robots to explore an unknown environment can clearly provide significant improvements in terms of robustness and efficiency to the whole system. Robots should coordinate their activities to explore as fast as possible the environment while avoiding conflicts (e.g., two robots trying to reach the same position in the environment). While several mechanisms, such as task assignment, can be successfully used for coordination, a main problem to solve, in the context of rescue scenarios, is to deal with possible lack of direct communication among robots and between robots and the human operator. Such problem can be addressed, by considering the maintenance of communication among the team as an objective for the coordinated exploration strategy. This is, in general, an objective that is opposite to the goal of the exploration. In fact, the exploration is usually targeted towards spreading the robots as much as possible in the environment to gather all possible information. While our approach is able to deal with multi-objectives exploration, further investigation is required to provide an efficient and general solutions to combine this two particular conflicting objectives.

Another very interesting research line, is to use *contextual* information in the exploration strategy. Contextual information is, in general, domain specific information which can be used inside a general process to have better performance. For example, for the exploration task, the use of contextual information can be exploited both for target selection and navigation. For target selection, contextual information can be used to escape a particular area that is too complex to be explored given the mobility capabilities of the robotic platform, or to avoid exploring an area where the probability to find interesting features is considered too low. As for navigation, the robot should be able to move quickly in areas that are already been explored and searched for features, while it needs to go slowly when it is looking for features (due, for example, to the computational time needed for classification algorithms). The use of contextual information is very interesting for complex domains such as the rescue scenario, and general solutions for *contextualization* of relevant information could result in more efficient and more robust systems.

6.3 Victim Detection

The presented approach is a first step towards human body detection in search and rescue missions. In order to evaluate the effectiveness of such system we have created a test data-set of stereo images including people in different poses and situations.

We plan to further develop the victim detection module in three ways: 1) develop and test different detection algorithms based on vision, like skin-color detection, face recognition, etc.; 2) investigate the use of different sensors: a thermic sensor and 2D laser range finder; 3) combine the results of such detection algorithms using a boosting approach (e.g., AdaBoost).

We also intend to proceed with a more systematic analysis of the performance of the developed methods, extending the data set to other relevant situations for rescue scenarios.

References

1. Carmen project, `www-2.cs.cmu.edu/∼carmen/`.
2. S. Balakirsky, C. Scrapper, S. Carpin, and M. Lewis. Usarsim: providing a framework for multi-robot performance evaluation. In *Proceedings of the International Workshop on Performance Metrics for Intellingent Systems (PerMIS)*, 2006.
3. A. et al. Birk. RoboCupRescue - Robot League Team IUB Rescue, Germany. In *RoboCup 2006: Robot Soccer World Cup XI*, 2006.
4. Andreas Birk, Kausthub Pathak, Soeren Schwertfeger, and Winai Chonnaparamutt. The IUB Rugbot: an intelligent, rugged mobile robot for search and rescue operations. In *IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR)*. IEEE Press, 2006.
5. James Bruce and Manuela Veloso. Real-time randomized path planning for robot navigation. In *Proceedings of IROS-2002, Switzerland, October 2002*, 2002.
6. W. Burgard, D. Fox, and Thrun S. A probabilistic approach to concurrent mapping and localization. *Machine Learning*, 1998.
7. D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi. Autonomous navigation and exploration in a rescue environment. In *Proc. of the 2nd European Conference on Mobile Robotics (ECMR)*, pages 110–115, Edizioni Simple s.r.l., Macerata, Italy, September 2005. ISBN: 88-89177-187.
8. D. Calisi, A. Farinelli, L. Iocchi, D. Nardi, and F. Pucci. Multi-objective autonomous exploration in a rescue environment. In *Proc. of IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*, 2006.
9. A. Carbone, G. Ugazio, A. Finzi, and F. Pirri. Robocuprescue - robot league team alcor. In *RoboCup 2004: Robot Soccer World Cup VIII*, 2004.
10. J.A Castellanos, J.M.M. Montiel, J. Neira, and J.D. Tardos. The spmap: a probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automatics*, 1999.
11. E. Feron E. Frazzoli, M.A. Dahleh. Real-time motion planning for agile autonomous vehicles. In *2000 AIAA Conf. on Guidance, Navigation and Control.*, 2000.
12. A. Eliazar and R. Parr. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, 2003.
13. Jianping Fan, David K.Y. Yau, Ahmed K. Elmagarmid, and Walid G. Aref. Automatic image segmentation by integrating color-edge extraction and seeded region growing. *IEEE Transaction on image processing*, 2001.
14. A. Farinelli, G. Grisetti, and L. Iocchi. Design and implementation of modular software for programming mobile robots. *International Journal of Advanced Robotic Systems*, 3(1):37–42, March 2006. ISSN 1729-8806.
15. Bernd Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.
16. Shahram Bahadori Ghouchani. *Human Body Detection in Search and Rescue Missions*. PhD thesis, University of Rome 'La Sapienza', Dipartimento Di Informatica e Sistemistica, 2006.
17. Héctor H. González-Baños and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *I. J. Robotic Res.*, 21(10-11):829–848, 2002.
18. G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.
19. G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Speeding up rao blackwellized slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 442–447, Orlando, FL, USA, 2006.
20. J.S. Gutmann and K Konolige. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE Interational Symposium on Computational Intellignece in Robotics and Automation*, 2000.

21. D. Haehnel, W. Burgard, D. Fox, and S. Thrun. An highly efficient algorithm for generating maps of large scale cyclic environments from raw laser range measurements, 2003.
22. *Proceedings of IEEE/RSJ IROS 2006 Workshop on Multi-objective Robotics (IROS-MOR 2006)*, Beijing, China, October 2006.
23. M.W. et al. Kadous. RoboCupRescue - Robot League Team CASualty, Australia. In *RoboCup 2006: Robot Soccer World Cup XI*, 2006.
24. L. Kavraki and J. Latombe. Probabilistic roadmaps for robot path planning. In *Practical Motion Planning in Robotics: Current Approaches and Future Challenges*, pages 33–53. K.G. and A.P. del Pobil, 1998.
25. A. Kleiner, C. Dornhege, R. Kuemmerle, M. Ruhnke, B. Steder, B. Nebel, P. Doherty, M. Wzorek, P. Rudol, G. Conte, S. Durante, and D. Lundstrom. Robocuprescue - robot league team rescuerobots freiburg (germany). In *RoboCup 2006 (CDROM Proceedings), Team Description Paper, Rescue Robot League*, Bremen, Germany, 2006. (winner of the 1st place of the autonomy competition).
26. A. et al. Kleiner. RoboCupRescue - Robot League Team Rescue Robots Freiburg, Germany. In *RoboCup 2006: Robot Soccer World Cup XI*, 2006.
27. K. Konolige. Small vision systems: Hardware and implementation. In *Proc. of 8th International Symposium on Robotics Research*, 1997.
28. J. Kuffner and S. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000), San Francisco, CA, April 2000.*, 2000.
29. J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publisher, 1991.
30. S. LaValle and J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE International Conf. on Robotics and Automation*, pages 473–479, 1999.
31. K. et al. Lingemann. RoboCupRescue - Robot League Team Deutschland1, Germany. In *RoboCup 2006: Robot Soccer World Cup XI*, 2006.
32. F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *CVPR94*, pages 935–938, 1994.
33. F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping, 1997.
34. M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.
35. P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1180–1187, Orlando, FL, USA, 2006.
36. Andreas Nchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6d slam – mapping outdoor environments. In *Proc.s Intl. Workshop on Safety, Security and Rescue Robotics (SSRR)*, Gaithersburg, MD, USA, 2006.
37. Andreas Nchter, Kai Lingemann, Joachim Hertzberg, Hartmut Surmann, Kai Pervlz, Matthias Hennig, K. R. Tiruchinapalli, Rainer Worst, and Thomas Christaller. Mapping of Rescue Environments with Kurt3D. In *IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR)*. IEEE Press, 2005.
38. F. et al Pirri. RoboCupRescue - Robot League Team Alcor, Italy. In *RoboCup 2006: Robot Soccer World Cup XI*, 2006.
39. R. Pito. A sensor based solution to the next best view problem, 1996.
40. G. Polverari, D. Calisi, A. Farinelli, and D. Nardi. Development of an autonomous rescue robot within the USARSim 3d virtual environment. In *Proceedings of RoboCup Symposium 06*, 2006. to appear.
41. Triebel R., P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.
42. J. H. Reif. Complexity of the mover's problem and generalization. In *Proc. 20th IEEE Symp. on Foundations of Computer Sciences (FOCS)*, pages 421–427, 1979.

43. Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios. Probabilistic cooperative localization and mapping in practice. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*, LasVegas, USA, 2003.
44. M. Singh and D.D Seyranian, G. and. Hoffman. Parsing silhouettes: The short-cut rule. *.Perception and Psychophysics, 61, 636-66*, 1999.
45. C. Stachniss, G. Grisetti, D. Haehnel, and W. Burgard. Improved rao-blackwellized mapping by adaptive sampling and active loop-closure. In *In Proc. of the Workshop on Self-Organization of AdaptiVE behavior (SOAVE)*, 2004.
46. D. Styrstrom and M. Holmqvist. RoboCupRescue - Robot League Team RFC Uppsala, Sweden. In *RoboCup 2006: Robot Soccer World Cup XI*, 2006.
47. S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. to appear.
48. B. Yamauchi. A frontier based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, July 10-11, 1997.*, 1997.
49. Liang Zhao. *Dressed human modeling, detection, and parts localization.* PhD thesis, The Robotic Institute Carnegie Mellon University,Pittsburgh, 2001.
50. V. A. Ziparo and L. Iocchi. Petri net plans. In *Proc. of Fourth International Workshop on Modelling of Objects, Components, and Agents (MOCA)*, pages 267–290, Turku, Finland, 2006. Bericht 272, FBI-HH-B-272/06.