RoboCup 2012 Rescue Simulation League Winners

Francesco Amigoni[1], Arnoud Visser[2], and Masatoshi Tsushima[3]

[1] Politecnico di Milano, Milano, Italy
francesco.amigoni@polimi.it
[2] University of Amsterdam, Amsterdam, The Netherlands
a.visser@uva.nl
[3] Ritsumeikan University, Shiga, Japan
is0077er@ed.ritsumei.ac.jp

Abstract. Inside the RoboCup Rescue Simulation League, the mission is to use robots to rescue as many victims as possible after a disaster. The research challenge is to let the robots cooperate as a team. This year in total 15 teams from 8 different countries have been active in the competitions. This paper highlights the approaches of the winners of the virtual robot competition, the infrastructure competition, and the agent competition.

1 Introduction

The RoboCup Rescue Simulation League consists of three competitions:

The **Virtual Robot competition** has the goal to study how a team of robots can work together to get as fast as possible a situation assessment of a devastated area which allows first responders to enter the danger zone well informed. The simulation of the robots is realistic enough to apply the same algorithms to real rescue robots.

The **Infrastructure competition** is a prize to stimulate the innovation factor and the impact of the competition. Progress inside the RoboCup Rescue Simulation League can only be made when each year the challenge gets harder. This can be accomplished by scaling the simulation environment up (larger disaster areas, more agents) or by including more realism into the simulation models. The Infrastructure competition is meant to foster innovation of models and components inside the simulation environment.

The **Agent competition** consists of a simulation platform which resembles a city after an earthquake. In this environment intelligent agents can be spawned, which influence the cause of events in the simulation. The agents have the role of police forces, fire brigades, and ambulance teams.

This paper presents the winner teams of the three competitions within the RoboCup 2012 Rescue Simulation League.

2 Virtual Robot Competition Winner Team PoAReT

PoAReT (*Politecnico di Milano Autonomous Robotic Rescue Team*) won the Virtual Robot competition of the Rescue Simulation League at RoboCup 2012. The PoAReT system is developed by six MSc students in Computer Engineering at the Politecnico di Milano. Full information about the team, including a

X. Chen et al. (Eds.): RoboCup 2012, LNAI 7500, pp. 20–35, 2013.

link to the source code and the list of members with their roles is available at http://home.dei.polimi.it/amigoni/research/PoAReT.html. In the following sections, we overview the PoAReT system architecture and summarize the most interesting scientific results obtained during the competition.

2.1 System Architecture

This section outlines the main features of the PoAReT system, as reported in [1], to which the reader is referred for further information. In developing PoAReT, we push along the *autonomy* axis, attempting to equip the robotic system with methods that enable its autonomous operation for extended periods of time. At the same time, the role of human operator is not neglected, but is empowered by the autonomous features of the system.

Besides the base station, our PoAReT system is composed of mobile platforms (usually the Pioneer All Terrain robot P3AT), each equipped with laser range finders, sonars, and a camera. Laser range finders are used to build a geometrical map of the environment that is represented with two sets of line segments. The first set contains the line segments that represent (the edges of) perceived obstacles. The second set contains the line segments that represent the *frontiers*, namely the boundaries between the known and the unknown portions of the environment.

The main cycle of activities of the PoAReT system is: (a) building a geometrical map of the environment composed of line segments, (b) selecting the most convenient frontiers to reach, and (c) coordinating the allocation of robots to the frontiers. A distinguishing feature of our system is that it can maintain a *semantic map* of the environment that labels areas of the geometrical map with human-like names, like 'room' or 'corridor'. At the same time, the system performs the detection of victims on the basis of the images returned by the onboard cameras and the interaction with the human operator via the user interface.

The architecture of our system is organized in two different types of processes, one related to the base station and one related to the mobile robots, to have a clear separation between their functionalities. Fig. 1 shows the PoAReT system architecture.

The base station embeds the user interface module. The base station process can spawn new robots in the USARSim environment [2]: for each robot, a new independent process is created and started. The processes of the base station and of the robots communicate only through WSS [3] and do not share any memory space, as required by the rules for the competition. A distance vector routing protocol [4] is implemented to deliver messages. Although in principle there is no need to maintain a direct connection between robots and base station (robots explore autonomously and, when connection is active, they can report to the base station and share collected information with other robots), the routing protocol maintains indirect connectivity between robots and base station in order to extend the operative range of the human operator.

The PoAReT User Interface (UI) allows a single human operator to control a relatively large group of robots in an easy way. It displays data to the user and accepts commands from the user to control the spawned robots. It reduces the workload of the operator and increases her/his situation awareness. These two objectives are reached by our UI through a mixed-initiative approach [5]. The PoAReT UI allows a single

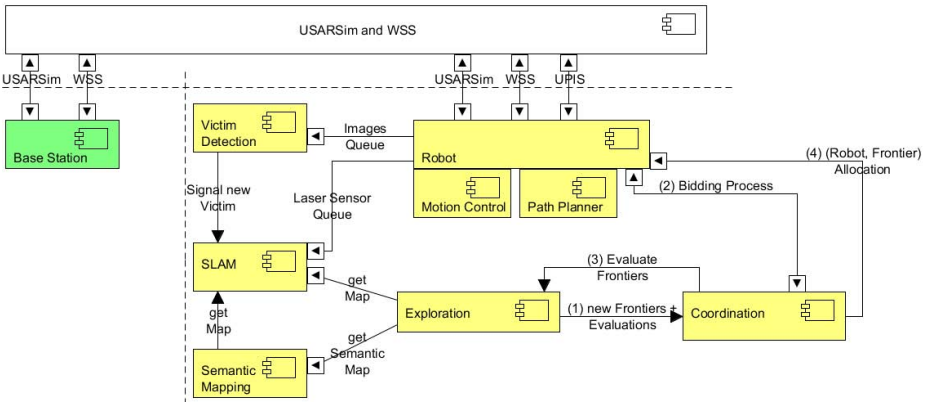


Fig. 1. PoAReT system architecture. The base station module is in green, while the mobile robot modules are in yellow.

operator to control the system by issuing high level commands to robots, like "explore along a direction", by controlling a single robot using waypoints, and by directly teleoperating the robot manually. The UI is also able to filter notifications arriving from the other modules, based on the operator's preferences, past behaviour, and situation parameters.

The robot process is structured in seven different modules, each one related to a high-level functionality: motion control, path planning, SLAM, semantic mapping, exploration, coordination, and victim detection. Almost all of these modules are threads that communicate through a queue system. The main functionalities of the above modules are described in the following.

First, we briefly discuss the motion control module, which is straightforward, given the locomotion model of P3AT, and the path planning module. Path planning is invoked to reach a position with a path that lies entirely in the known space (e.g., the position can be a point on a frontier between known and unknown space). The algorithm we use is a variant of RRT [6].

In our team, the simultaneous localization and mapping (SLAM) problem is tackled by adopting a feature-based method similar to that described in [7]. The SLAM module associates the line segments of a laser scan (points of a scan are approximated with line segments by using the split and merge algorithm [8]) to the the linear features in the map, with respect to distance measures, such as those described in [9, 10]. Then, the module executes an Iterative Closest Line (ICL) algorithm (like [10]) with constraints on the maximum rotation and on the maximum translation to align the scan and the map. All the line segments of a scan are added to the map; periodically a test is carried out to determine whether there is enough evidence to support the hypothesis of two previously associated line segments being in fact the same; if so, they are merged.

The semantic mapping module performs a semantic classification of places and works in parallel with the SLAM module. This module takes as input the line segment map of an indoor environment (updated by the SLAM module) and tries to extract more information than the basic geometrical features, exploiting prior knowledge on

the typical structure of buildings. Our approach aims at extending that presented in [11] and [12] to line segment maps. The mapped area is divided into single rooms, identifying the area that belongs to each room and the doorways that divide the rooms. With this information, the space portion marked as *room* is divided into different parts representing every single room separately. Later, each room is classified according to its own characteristic, as a *small room*, a *large room*, or a *corridor*.

The exploration module selects new frontiers to explore, in order to discover the largest possible amount of the environment within the time allowed in the competition. This module evaluates the frontiers by assigning them utilities and, finally, calls the coordination module to find an allocation of robots to the frontiers. We employ an exploration strategy that exploits the geometrical and semantic information gathered by the robots. We take inspiration from [13], where the authors achieve a good exploration performance by distinguishing if the robot is in a hallway or in a room. In our system, we integrate this semantic information into a framework, called Multi-Criteria Decision-Making (MCDM), that is described in [14].

The coordination module is responsible of allocating tasks to the robots. The mechanism we use is market-based and sets up auctions in which tasks (i.e., frontiers to reach) are auctioned to robots [15]. These market-based mechanisms provide a well-known mean to bypass problems like unreliable wireless connections or robot malfunctions.

Finally, the victim detection module is responsible for searching victims inside the competition environment. It works by analysing images coming from the robots' cameras and classifying them according to the presence or absence of victims. In the first case, the victim detection module signals the human operator. We have chosen to implement a skin detector using HSV (Hue, Saturation, Value) color space, followed by a version of the Viola-Jones algorithm [16], a well-known image analysis method already used by many teams in previous editions of the competition.

2.2 Discussion on Competition Results

Besides the good performance that allowed the PoAReT team to win the Virtual Robot competition, some potentially interesting scientific outcomes have been obtained, as discussed in this section.

Firstly, the geometrical maps built by the system and representing the environments of the competition are of good quality, demonstrating the viability of using line segments to represent indoor environments. For example, Fig. 2 shows the geometrical map built by the PoAReT system for the environment of the Day 2 of the competition. Note that, in this run, the maps built by different robots are not merged together, in order to reduce computational burden (this is why some obstacles are represented by multiple aligned line segments). The structure of the environment is represented quite well for understanding by human operator and, importantly, using a limited amount of data (each line segment can be naïvely represented by four numbers). It is also interesting noting that on the Day 3 of the competition, a non-regular indoor environment has been used with several obstacles and with different (vertical) levels. In this case, the segment-based approach has not been much effective to represent the environment.

Second, the availability of a semantic map has been exploited for improving path planning. In particular, the identified *doorways* (i.e., openings between two rooms; an

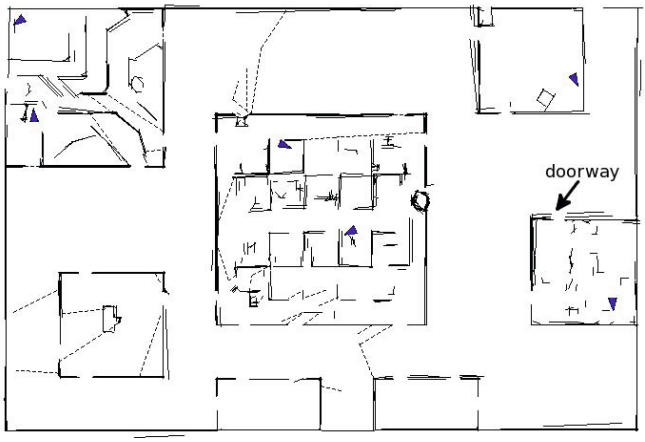


Fig. 2. The geometrical map built by the PoAReT system on Day 2 of the competition. The solid line segments represent obstacles, the dashed line segments represent the frontiers, and the blue triangles represent the positions of the robots.

example is shown on the right of Fig. 2) have been used by the path planner to set waypoints such that a robot crosses a doorway by heading perpendicularly to the line segment representing it. In this way, the robots of our system have been able to cross narrow doors, significantly enhancing their path planning capability.

Third, the high level of autonomy exhibited by the PoAReT system has allowed to explore structured indoor environments very quickly. For example, the results of the first three days of the competition show that PoAReT found 11 victims in 41 minutes, while the next two teams found the same number of victims in 57 and 59 minutes, respectively.

Finally, the autonomy of the PoAReT system does not reduce the role of human operator. In some runs at the competition, Kenaf and AirRobot mobile platforms have been used. Kenafs have been controlled using a controller that is very similar to that of P3AT (without fully exploiting the Kenaf abilities), while AirRobots have been teleoperated manually. The increased workload for the human operator has been compensated by the ability of the system to reach areas (e.g., requiring to climb a stair) that P3ATs cannot reach. In general, teams mainly composed of P3ATs and of some Kenafs and AirRobots showed a good level of adaptability to environments, autonomous behavior, and performance.

3 Infrastructure Competition Winner Team UvA Rescue

The University of Amsterdam is active in the Rescue Simulation League with the UvA Rescue team since 2003 [17, 18]. For several years it had a close cooperation with Oxford University [19]. On several occasions the team contributed to the infrastructure of the competition [20–24]. The system presented at the 2012 Infrastructure competition was developed by a master student in Artificial Intelligence [25].

3.1 The Context

It is well known [26–28] that an aerial robot is a valuable member of a robot team. Several groups indicated the usage of aerial robots in their teams [29–31], as illustrated in Fig. 3, but without a map it is difficult to coordinate the action between the teammembers [28]. Because of the limited payload the aerial robots can carry, it is difficult to equip those robots with range scanners. Without range scanners it is difficult for aerial robots to navigate [32] and to create a map of the environment [33].

Fig. 3. An early example of the usage of an aerial robot in robot rescue team (courtesy [31]).

Nowadays, small quadrotors with on-board stabilization like the Parrot AR.Drone can be bought off-the-shelf. These quadrotors make it possible to shift the research from basic control of the platform towards applications that make use of their versatile scouting capabilities. Possible applications are surveillance, inspection, and search and rescue. The Parrot AR.Drone is attractive as platform, because it is stabilized both horizontally and vertically. Horizontal movement is reduced based on the images of the bottom camera, while the altitude is maintained based on the signal of a downlooking sonar sensor. Still, the limited sensor suite and the fast movements make it quite a challenge to fully automate the navigation for such platforms. One of the prerequisites for autonomous navigation is the capability to make a map of the environment.

Once such a map exists, a team of micro aerial vehicles could be used to explore an area like a city block. The map is needed to coordinate the actions between the team members. After a disaster one could not rely on prior satellite maps, part of the job of the rescue team is to do a situation assessment and an estimation of damage (roads blocked, buildings on fire, locations of victims visible from the sky).

In the paper presented at the Infrastructure competition [34] a method is described that shows how such a visual map can be built. More details can be found in [25]. To summarize; the visual map consists of a feature map which is built based on storing the

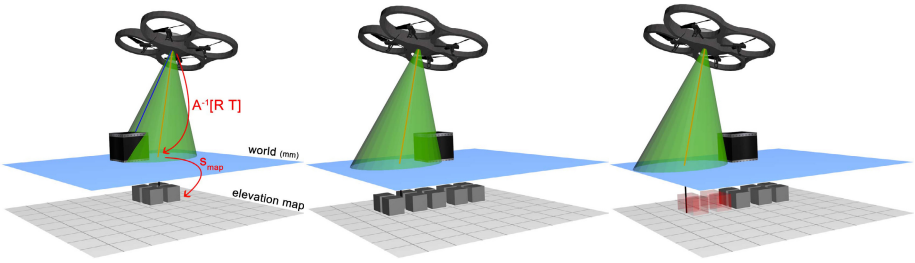
most distinguishable SURF features on a grid. This map can be used to estimate the movement from the AR.Drone on visual clues only, as described in previous work [35]. In this paper the focus is on an extension of the previous method, an experimental method [25] to create an elevation map by combining the feature map with ultrasound measurements. This elevation map is combined with textures stored on a canvas and visualized in real time. An elevation map is a valuable asset when the AR.Drone has to explore unstructured terrain, which is typically the case after a disaster (an urban search and rescue scenario).

More details about how the feature map can be used to localize the AR.Drone can be found in [35]. What is really innovative is how the 2D feature map is extended with a method to build an elevation map based on sonar measurements, which was published in the paper for the Infrastructure competition [34]. This paper demonstrates how the elevation mapping method was validated with experiments.

3.2 Elevation Mapping Method

An elevation map can be used to improve navigation capabilities of both aerial and ground robots. For example, ground robots can use elevation information to plan routes that avoid obstacles.

The elevation information is stored in a grid that is similar to the feature map described in [35]. For each ultrasound distance measurement, elevation δ_t is computed and stored in the grid cell that corresponds to the world coordinates where a line perpendicular to the AR.Drone body intersects the world plane. These world coordinates are the position where the center of the ultrasound sensor's cone hits the floor. Because the exact size of an object is unknown, the elevation is written to all grid cells within a radius $\gamma_{elevationRadius}$ around the intersection point. This process is visualized in Figs. 4a and 4b.



(a) Obstacle enters range (b) Obstacle in range (c) Obstacle out of range

Fig. 4. Overview of the elevation map updates. The green cone indicates the range of the ultrasound sensor. The red line inside the cone represents the center of the cone, perpendicular to the AR.Drone body. In 4a and 4b an elevation is measured. All grid cells within a radius $\gamma_{elevationRadius}$ around the center of the cone (red line) are updated to store the measured elevation. 4c describes the refinement step. When no elevation is measured, all grid cells within the cone (red cubes) are reset to zero elevation.

This approach may lead to cases where the size of an obstacle is overestimated in the elevation map, as can be seen in Fig. 4b. Therefore, an additional refinement step was added to the elevation mapping method. If no elevation is measured ($\delta_t \approx 0$), it can be assumed there is no obstacle inside the cone of the ultrasound sensor. Using this assumption, all grid cells within the cone can be reset to zero elevation and locked to prevent future changes. This refinement step is visualized in Fig. 4c. The radius of the cone is computed using the following equation:

$$r = \tan(\alpha_{ultrasound} \times z_{sensor}) \tag{1}$$

where r is the radius of a cone of height z_{sensor} and $\alpha_{ultrasound}$ is the opening angle of the ultrasound sensor.

3.3 Elevation Mapping Results

The elevation mapping approach has been validated with several experiments. As shown in Fig. 5, the AR.Drone is able to estimate the height and length of two obstacles (a grey and blue box). The brown box is not detected due to its limited height. Therefore, the measured (ultrasound) acceleration is insufficient to trigger an elevation event. As expected, the ramp was not detected. The gradual elevation change does not produce a significant acceleration.

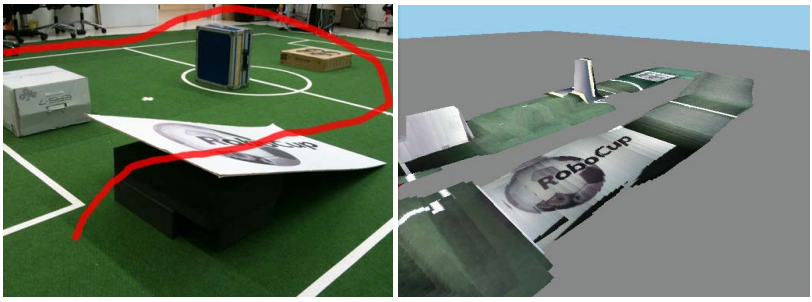


Fig. 5. Elevation map of a flight over several obstacles. On the left the experimental setting (including 4 obstacles) is visible, augmented in red with the path of AR.Drone. On the right the resulting map is displayed, with at the back to elevated areas, representing the white and blue box. For convenience the grid cells of the elevation map are colored with the texture at that point as perceived by the downlooking camera of the AR.Drone.

Elevation changes are detected using a filtered second order derivative (acceleration) of the sonar measurement z_{sensor}, as illustrated in Fig. 6.

Obstacles that enter or leave the range of the ultrasound sensor result in sudden changes in ultrasound distance measurements. These changes are detected when the second order derivative exceeds a certain threshold $\gamma_{elevationEvent}$ and an elevation event is triggered. The threshold $\gamma_{elevationEvent}$ was carefully chosen such that altitude corrections performed by the AR.Drone altitude stabilization are not detected as being elevation events. An elevation event ends when the sign of the second order derivative

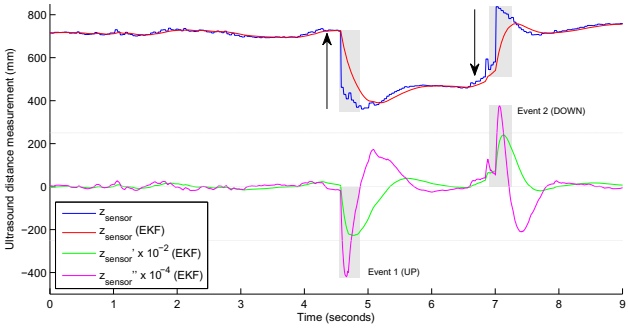


Fig. 6. Response of the ultrasound sensor when flying over an object of approximately $(60, 60, 40)$ mm. The light gray lines indicate the threshold $\gamma_{elevationEvent}$ and null-line. When the second order derivative (magenta line) exceeds the threshold, an event is started (light gray rectangle). An event ends when the derivative swaps sign. Each arrow indicates the change in elevation caused by the event. The first event increases the elevation when entering an object and the second event decreases the elevation when leaving the object. Between both events, the AR.Drone performs an altitude correction, as can be seen by the relatively slow increase of the distance. This increase is not informative about the elevation and is ignored by the elevation mapping method.

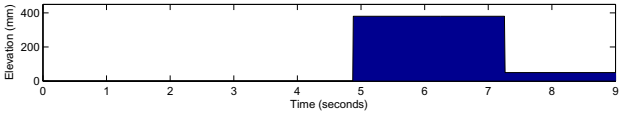


Fig. 7. Elevation δ below the AR.Drone over time. The elevation increases to approximately 40 cm when flying above an obstacle. The elevation is decreased when the obstacle is out of the ultrasound sensor's range. There is a small error between both elevation events, resulting in a small false elevation (± 50 mm) after the AR.Drone flew over the obstacle and is flying above the floor again.

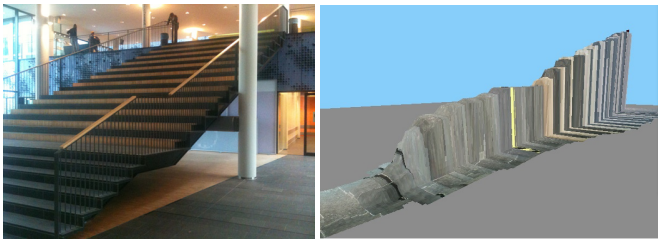


Fig. 8. Photo and map of a large stair at our university which is traversed by the AR.Drone. The depth of each step is 30 cm and the height of each step is 18 cm. The total height of the stair is 480 cm.

switches. This happens when the AR.Drone altitude stabilization starts to change the absolute altitude to compensate for the change in measured altitude. Now, the elevation change can be recovered by subtracting the measured distance at the end of the elevation event from the measured distance before the elevation event was triggered, as illustrated in Fig. 7.

In a final experiment, the AR.Drone flew with a constant speed over a large stair (Fig. 8). The depth of each step is $30\ cm$ and the height of each step is $18\ cm$. The total height of the stair is $480\ cm$. After the stair is fully traversed by the AR.Drone, the estimated elevation is compared against the actual height of the stair.

After fully traversing the stair, the measured elevation is $313\ cm$ and the error is $\frac{480-313}{480} \times 100 = 35\%$. The shape of the measured elevation corresponds with the shape of the stair. However, the approach underestimates the elevation. When traversing the stair, the AR.Drone's altitude stabilization increases the altitude smoothly, which causes a continuous altitude increase. Therefore, the observed altitude difference within an elevation event is smaller than the actual altitude difference caused by an object. Another explanation of the underestimation is the error in the triggering of elevation events (due to thresholds). When an elevation event is triggered at a suboptimal timestamp, the full altitude difference is not observed.

3.4 Summary

The experiments demonstrate what is possible for rapid development when a realistic simulation environment for the AR.Drone is available. The simulation model of the AR.Drone is made publicly available[1] inside the USARSim environment. The validation of this model was described in [35]. We hope that the availability of such a model inside the infrastructure of the RoboCup Rescue Simulation League will contribute in attracting researchers to develop advanced algorithms for such a system.

The mapping method described in [34] is able to map areas visually with sufficient quality for both human and artificial navigation purposes. Both the real and simulated AR.Drone can be used as platform for the mapping algorithm. The visual map created by the simulated AR.Drone contains fewer errors than the map of the real AR.Drone. The difference can be explained by the variance in the lighting conditions encountered by the real AR.Drone. Notice that the USARSim environment is based on a commercial game engine (Unreal Tournement) which already simulates many lighting conditions (e.g., shadows, reflections) quite realistically.

Earlier work [35] shows that the visual map can be used to localize the AR.Drone and significantly reduce the error of the estimated position when places are revisited. Important for a good visual map is that sufficient information is available on the ground; for instance when long straight lines in a gym are followed the travelled distance is underestimated. To conclude; visual mapping is an important capability to scale up the robot team to the level where they can explore a city block, which is close to the current challenge in the Agent competition of the RoboCup Rescue Simulation league.

[1] `http://sourceforge.net/apps/mediawiki/usarsim/index.php?title=Aerial_Robots#AR.Drone`

4 Agent Competition Winner Team Ri-one

This section describes the RoboCup 2012 Rescue Simulation League Agent competition champion team, Ri-one. Our team consists of four students from the Faculty of Information Science and Engineering at Ritsumeikan University. The basic structure of our agent is divided into models and skills.

4.1 Models

The agents have to make informed decisions. The decisions are based on information which is embedded in a World Model. The World Model not only stores a priori and perceived information, but also makes inferences to allow more efficient actions of the agents.

Applying Point of Visibility Navigation Graph. When the agents have the intention to move somewhere in the simulated city, they must send their path as a list of areas (cells) to the server. The agents get the map without any obstacle at the start of simulation, and receive information about nearby open space and obstacles via their sensors at each step. Information given to agents about open space has the form of connected two-dimensional closed shapes, as illustrated in Fig. 9. Obstacles blocking routes have also the closed shapes. With this information, the agents must plan their path in limited time. The path planning is typically performed with a graph search algorithm [36]. The map cannot be converted directly into a graph of connected nodes because the shapes of the cells are not always a convex polygon. The map has to be converted to a new graph which is called a Point of Visibility Navigation Graph when the simulation starts. Therefore, we developed the following method to generate this graph automatically.

In order to generate the graph, we have to consider the relation between nodes and areas defined by the closed shapes. First of all, nodes can be defined as the points (do not need to be centers) within the areas. This is necessary since the path to move along is determined by the list of areas to be visited. However, when connecting the points whose areas are adjacent does not guarantee a collision free path, as shown in red in Fig. 9. Hence, we added additional nodes to the graph to solve this problem. These intermediate nodes are placed on the boundaries of adjacent areas and not explicitly shown in Fig. 9. Therefore, the end result is the undirected bipartite graph which has two kinds of nodes, terminal nodes and non-terminal nodes. Terminal nodes can be used as a start- or end-point of a path, and they must inside an area. On the other hand, non-terminal nodes are intermediate points, on the edges between areas and cannot be a start or an end node to any path.

The algorithm to generate Point of Visibility Navigation Graph has the following pseudo code:

1. Set a terminal node to every area.
2. List all pairs of adjacent areas.

3. For all pairs of areas, define the middle point of the shortest line segment between them (refer to two buildings as A and B, edge of A to B and B to A) as non-terminal node.
4. Relate terminal nodes and non-terminal nodes mutually, according to their visibility.

This method creates new traversable edges according to the visibility of nodes. Fig. 10 shows these lines. With the Point of Visibility Navigation Graph the agents can efficiently perform collision free path planning.

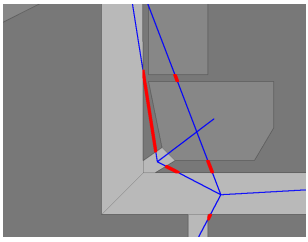


Fig. 9. Connecting areas by connecting the center of the cells. Blue and red line segments represent traversable and non-traversable paths, respectively.

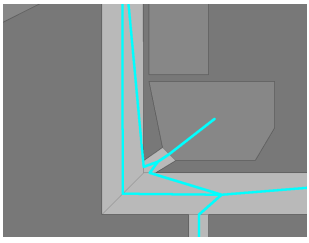


Fig. 10. Generated Point of Visibility Navigation Graph. Cyan line segments represent edges.

Estimating Fires. This estimation makes two assumptions. The first assumption is that an influence of the building's temperature depends on the temperature of another building on fire. The other assumption is that heat spreads in the form of concentric spheres centered on the burning building. When a building whose temperature is t affects another building r meters away from its center, a surface area of sphere with radius r is defined to be S, and a coefficient defined to be k, the influence I satisfies the following relation:

$$\oint_S I dS = kt \tag{2}$$

This I is the influence within the sphere. Then, the angle formed by the lines from the burning building to the intercept of the affected building and affected edge is defined to be θ. An influence I of an infinitesimal surface of the sphere is defined as follows:

$$I = \sin\theta \frac{kt}{4\pi r^2} \tag{3}$$

Fig. 11 shows this idea. This will make it possible to estimate the probability that an invisible building is on fire or not, by calculating the value of I in relation to the temperature of that building.

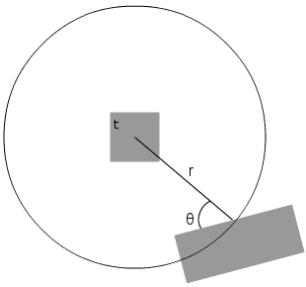


Fig. 11. Influence of temperature t of a building on another building at distance r and angle θ

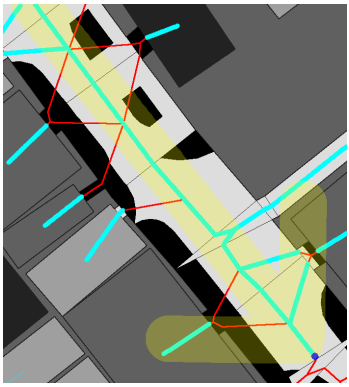


Fig. 12. The result of using Point of Visibility Navigation Graph. The traversable edges are represented by cyan line segments and the non-traversable edges are represented by red line segments.

4.2 Agent Skills

Agents select the best actions from the World Model in order to carry out the operations. The main idea of success of Team Ri-one is the Police Force's skill.

Police Force. Police Forces clear the obstacles caused by the disaster. They must clear the obstacles efficiently to help actions of other agents including Ambulance Teams and Fire Brigades. Therefore, police forces have to choose an obstacle and decide the amount to clear. In order to solve this problem, we use Point of Visibility Navigation Graph from Section 4.1 to decide the obstacle which the police forces will clear. First, a police force computes the shortest path to a target entity without considering obstacles. Secondly, they consider the line segments which compose the shortest path in cleared range. Since each line segment belongs to a single area because of the way the graph has been defined, it is evaluated whether intersections of the line segments and the shapes of all obstacles expanded by a fixed amount exist or not. When an intersection exists on the path, they clear the obstacle. When an intersection does not exist on the path, they move along the path. After that, if an intersection appears on the path, they clear the obstacle likewise when they discover the new obstacle. By repeating this method, they can reach the target entity without clearing obstacles which do not need to be cleared. Fig.12 shows the result of the algorithm application.

4.3 RoboCup 2012 Rescue Simulation League Agent Competition Results

In RoboCup 2012, the Ri-one team won the competition. The success was based on the reduction of unnecessary steps in the planning (for instance the improved Police Force's skill which ignores obstacles which do not have to be cleared) and on the prediction of the dynamics of the simulation (for instance the estimation of the spread

of a fire). The improved efficiency was enough to beat our competitors with a narrow margin[2]. The ZJUBase team from Institute of Cyber-Systems and Control, Zhejiang University, China, won second place, and the S.O.S team from Amirkabir University of Technology, Iran, won third place.

5 Conclusion

This paper gives a brief insight in the methods applied by the three winners of the RoboCup Rescue Simulation League. It demonstrates the variety of methods which have to be integrated to create a robot team which is able to cooperate to accomplish the mission to rescue as many victims as possible. The Rescue Simulation League is a competition which keeps on innovating. The DARPA organization has chosen robot rescue as the next challenge and it will be task of the RoboCup community to demonstrate the value of its benchmarks with relation to the scenario of the DARPA Challenge.

Acknowledgement. The PoAReT team gratefully thanks the Fondazione Banca del Monte di Lombardia for the financial support. The UvA research is partly funded by the EuroStars project 'SmartINSIDE' and the Dutch ICT Innovation Platform Cooperation Challenge 'SI4MS'.

References

1. Amigoni, F., Caltieri, A., Cipolleschi, R., Conconi, G., Giusto, M., Luperto, M., Mazuran, M.: PoAReT Team Description Paper. In: RoboCup 2012 CD (2012)
2. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: USARSim: a robot simulator for research and education. In: Proceedings of the International Conference on Robotics and Automation (ICRA 2007), pp. 1400–1405 (2007)
3. Pfingsthorn, M.: RoboCup rescue virtual robots: Wireless simulation server documentation. Technical Report, Jacobs University (2008)
4. Comer, D.: Internetworking with TCP/IP, vol. 1. Addison-Wesley (2006)
5. Wang, J., Lewis, M.: Human control for cooperating robot teams. In: Proc. HRI, pp. 9–16 (2007)
6. LaValle, A., Kuffner, J.: Rapidly-exploring random trees: Progress and prospects. In: Donald, B., Lynch, K., Rus, D. (eds.) Algorithmic and Computational Robotics: New Directions, pp. 293–308 (2001)
7. Garulli, A., Giannitrapani, A., Rossi, A., Vicino, A.: Simultaneous localization and map building using linear features. In: Proc. ECMR, pp. 44–49 (2005)
8. Nguyen, V., Gächter, S., Martinelli, A., Tomatis, N., Siegwart, R.: A comparison of line extraction algorithms using 2D range data for indoor mobile robotics. Autonomous Robots 23(2), 97–111 (2007)
9. Elseberg, J., Creed, R., Lakaemper, R.: A line segment based system for 2D global mapping. In: Proc. ICRA, pp. 3924–3931 (2010)
10. Li, Q., Griffiths, J.: Iterative closest geometric objects registration. Computers & Mathematics with Applications 40(10-11), 1171–1188 (2000)

[2] The detailed competition results can be found at:
`http://roborescue.sourceforge.net/`

11. Pronobis, A., Martinez Mozos, O., Caputo, B., Jensfelt, P.: Multi-modal semantic place classification. International Journal of Robotics Research 29(2-3), 298–320 (2009)
12. Martinez Mozos, O.: Semantic Labeling of Places with Mobile Robots. Springer (2010)
13. Stachniss, C., Mozos, O.M., Burgard, W.: Speeding up multi-robot exploration by considering semantic place information. In: Proc. ICRA, pp. 1692–1697 (2006)
14. Basilico, N., Amigoni, F.: Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. Autonomous Robots 31(4), 401–417 (2011)
15. Zlot, R., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Proc. ICRA, pp. 3016–3023 (2002)
16. Viola, P., Jones, J.J.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
17. Fassaert, M.L., Post, S.B.M., Visser, A.: The common knowledge model of a team of rescue agents. In: Proc. 1st International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster (2003)
18. Post, S.B.M., Fassaert, M.L., Visser, A.: The high-level communication model for multiagent coordination in the robocuprescue simulator. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 503–509. Springer, Heidelberg (2004)
19. de Hoog, J.: Role-Based Multi-Robot Exploration. PhD thesis, University of Oxford (2011)
20. Schmits, T., Visser, A.: An Omnidirectional Camera Simulation for the USARSim World. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 296–307. Springer, Heidelberg (2009)
21. Balaguer, B., Balakirsky, S., Carpin, S., Visser, A.: Evaluating maps produced by urban search and rescue robots: lessons learned from RoboCup. Autonomous Robots 27(4), 449–464 (2009)
22. Terwijn, B., Formsma, O., Dijkshoorn, N., van Noort, S., de Hoog, J., Out, N., Bastiaan, C., Visser, A.: Amsterdam Oxford Joint Rescue Forces: Community Contribution (2010) (published online)
23. Formsma, O., Dijkshoorn, N., van Noort, S., Visser, A.: Realistic Simulation of Laser Range Finder Behavior in a Smoky Environment. In: Ruiz-del-Solar, J. (ed.) RoboCup 2010. LNCS, vol. 6556, pp. 336–349. Springer, Heidelberg (2010)
24. van Noort, S., Visser, A.: Validation of the dynamics of an humanoid robot in USARSim. In: Proc. PerMIS (2012)
25. Dijkshoorn, N.: Simultaneous localization and mapping with the AR.Drone. Masters thesis, Universiteit van Amsterdam (2012)
26. Davids, A.: Urban search and rescue robots: from tragedy to technology. IEEE Intelligent Systems 17(2), 81–83 (2002)
27. Balakirsky, S., Carpin, S., Kleiner, A., Lewis, M., Visser, A., Wang, J., Ziparo, V.A.: Towards heterogeneous robot teams for disaster mitigation: Results and Performance Metrics from RoboCup Rescue. Journal of Field Robotics 24(11-12), 943–967 (2007)
28. Alnajar, F., Nijhuis, H., Visser, A.: Coordinated action in a Heterogeneous Rescue Team. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 1–10. Springer, Heidelberg (2010)
29. Pfingsthorn, M., Rathnam, R., Stoyanov, T., Nevatia, Y., Ambrus, R., Birk, A.: Jacobs Virtual Robot 2008 Team - Jacobs University Bremen, Germany. In: RoboCup 2008 CD (2008)
30. Calisi, D., Randelli, G., Valero, A., Iocchi, L., Nardi, D.: SPQR Rescue Virtual Robots Team Description Paper. In: RoboCup 2008 CD (2008)
31. Balaguer, B., Carpin, S.: UC Mercenary Team Description Paper: RoboCup 2008 Virtual Robot Rescue Simulation League. In: RoboCup 2008 CD (2008)

32. Visser, A., Nguyen, Q., Terwijn, B., Hueting, M., Jurriaans, R., van der Veen, M., Formsma, O., Dijkshoorn, N., van Noort, S., Sobolewski, R., Flynn, H., Jankowska, M., Rath, S., de Hoog, J.: Amsterdam Oxford Joint Rescue Forces - Team Description Paper - Virtual Robot competition - Rescue Simulation League - RoboCup 2010. In: RoboCup 2010 CD (2010)
33. Nguyen, Q., Visser, A.: A color based rangefinder for an omnidirectional camera. In: Balakirsky, S., Carpin, S., Lewis, M. (eds.) Proc. IROS Workshop on Robots, Games, and Research: Success stories in USARSim, pp. 41–48 (2009)
34. Dijkshoorn, N., Visser, A.: An elevation map from a micro aerial vehicle for urban search and rescue. In: RoboCup 2012 CD (2012)
35. Dijkshoorn, N., Visser, A.: Integrating sensor and motion models to localize an autonomous AR. Drone. International Journal of Micro Air Vehicles 3(4), 183–200 (2011)
36. Buckland, M.: Programming Game AI by Example. Wordware Publishing (2005)