The Design of a Fully Autonomous Robot System for Urban Search and Rescue

Yi Liu, Yuhua Zhong, Xieyuanli Chen, Pan Wang,
Huimin Lu, Junhao Xiao, Hui Zhang

College of Mechatronics and Automation,
National University of Defense Technology
Changsha, Hunan, China, 410073

lewis_nudt@foxmail.com, lhmnew@nudt.edu.cn
junhao.xiao@ieee.org, zhanghui_nudt@126.com

***Abstract* - Autonomous robots in urban search and rescue (USAR) have to fulfill several tasks at the same time: localization, mapping, exploration, object recognition, etc. This paper describes the whole system and the underlying research of the NuBot rescue robot for participating RoboCup Rescue competition, especially in exploring the rescue environment autonomously. A novel path following strategy and a multi-sensor based controller are designed to control the robot for traversing the unstructured terrain. The robot system has been successfully applied and tested in the RoboCup Rescue Robot League (RRL) competition and won the championship of 2016 RoboCup China Open RRL competition.**

Index Terms* - *USAR; autonomous navigation; RoboCup; exploration planner.

I. Introduction

A. Key Abilities of Autonomous USAR Robots

Due to natural disasters, terrorist activities and a variety of other incidents, disasters occur frequently around the world. Searching immediately to locate victims in collapsed buildings is the most effective way to save their lives. Rescue robots can be used to explore the unknown disaster environment which is dangerous for human beings, acquiring and sending back sensor data such as thermal and color videos. While these robots will remain mainly remote-controlled in the near future when used in real disaster sites, increasing the autonomy level is an efficient way to relieve the operator from the exhausting task of controlling the robot and may has the potential to vastly improve the capabilities of robots used for disaster response in the future.

Key abilities for fully autonomous rescue robots include building maps while localizing themselves, determining the next exploring target point, planning the path, following the path by motion control, as well as detecting and locating victims.

B. Overview of the RoboCup RRL Competition

The RoboCup Rescue Robot League (RRL) competition [1] simulates a rescue operation after an earthquake. The building partially collapses due to the earthquake. The incident commander in charge of rescue operations at the disaster site, fearing secondary collapses from aftershocks, asks for teams of robots to immediately search the interior of the building for victims. The mission for the robots and their operators is to find victims, obtain their situations, states, and locations, and then report their findings in a map of the building with associated victim data. The section near the building entrance appears relatively intact while the interior of the structure exhibits increasing degrees of collapses. A sample arena of the RoboCup RRL competition is shown in Fig. 1. Robots must negotiate and map the lightly damaged areas prior to encountering more challenging obstacles and rubble. This competition provides a standard test bed for robots that can autonomously explore unknown and unstructured environments.

Fig. 1 A sample arena of the RoboCup RRL competition.

After participating in the RoboCup RRL competition for almost ten years, we focus on increasing the robot autonomy level for Urban Search and Rescue (USAR) in recent years. In 2016, our team NuBot won the championship of the RoboCup China Open RRL competition with the ability to explore the complex and unknown rescue environment fully autonomously.

C. The Contribution of This Paper

The main contribution of this paper is to provide a successful autonomy solution for USAR robots especially in the rescue environment filled with uneven terrains with ramps.

D. The Arrangement of This Paper

In this paper we present recent advances in USAR robots, provide an overview of our autonomous robot system and present an efficient control strategy we use.

The remainder of the paper is organized as follows: Section II provides an overview of the robot hardware and software of our rescue robot, where Robot Operating System (ROS) has been used to build the software. The HectorSLAM algorithm and our custom designed LiDAR stabilizer are introduced in Section III. In Section IV, an autonomous exploration approach is discussed and a multi-sensor based controller is proposed to drive the robot to follow the trajectory in challenging terrains with high robustness. Section V presents how to detect and localize the simulated victims using thermal camera. Afterwards, we assess the robot performance in the rescue scenarios of our laboratory and the 2016 RoboCup China Open RRL competition. Finally Section VII concludes the paper.

II. System Overview

Many impressive results about the design of the robot system for USAR missions have been achieved, and various robots have been developed for USAR tasks. In early years, rescue robots were usually teleoperated by human operators. With robotic technology advances significantly, the robot's autonomy level has been improved greatly. We also designed our rescue robot system to realize fully autonomous USAR.

A. Hardware

Our robot uses a tracked platform, as shown in Fig. 2. The tracked robot with front and back sub-tracks (flippers) provides effective mobility, and it is the mostly common platform used in the RoboCup RRL competitions and also in the real rescue missions.

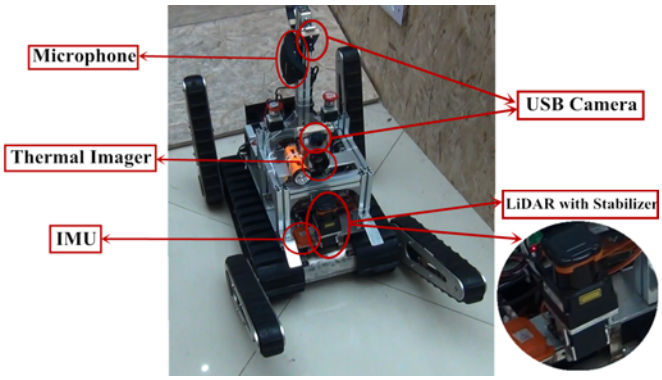


Fig. 2 NuBot participating the RoboCup China Open 2016 in Hefei.

In order to complete the USAR missions, the mobile robot must be equipped with an onboard computer and various sensors for mapping, navigation and victim detection.

The robot uses a state-of-the-art industrial grade computer from Beckoff. The computer (Intel Core i7) provides enough processing ability to deal with huge data and the robustness when traversing challenging terrains.

The robot is equipped with a Hokuyo UTM-30LX Light Detection And Ranging (LiDAR). The LiDAR is suitable for mobile robots because of its low power consumption and compact size. The field of view of the scanner is 270°, the scanning distance is 30m and the scanning frequency is 40Hz. The performance of Hokuyo UTM-30LX was evaluated in [2], where the results show that Hokuyo UTM-30LX can be used for distance measurement, and the quality of the acquired data is almost the same on different surfaces, colors and even under different illumination.

In order to measure the robot's pose when exploring in the unstructured and uneven terrains, a 6DOF inertial sensor, "Xsens MTI-100" has been integrated. MTI-100 is a miniature Inertial Measurement Unit (IMU) that outputs yaw angle with no drift, and provides a calibrated three-axis acceleration, angular velocity and magnetic field strength.

Visual perception is the most important source for victim detection. Therefore a pan-tilt-zoom camera is mounted on the tracked platform. Besides, a low-cost USB video camera and a Thermal Image Optris PI640 has been employed. The visual sensors are fused to detect and localize victims.

B. ROS-based Software

Robot Operation System (ROS) [3] is used to build the software for our rescue robot. It is the most popular robotic framework nowadays. It provides open source tools, libraries, and drivers for robotics researches and applications. ROS enables researchers to quickly and easily conduct experiments. The software architecture of our rescue robot is shown in Fig. 3.

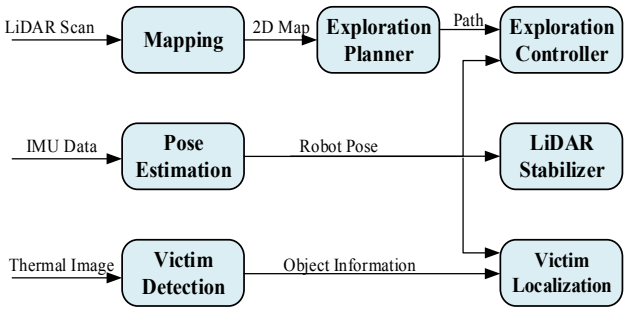


Fig. 3 The software architecture based on ROS. ROS nodes are represented by rectangles, topics by arrow-headed and services by diamond-headed lines. Services are originated at the service caller.

III. 2D SLAM with Active LiDAR Stabilizer

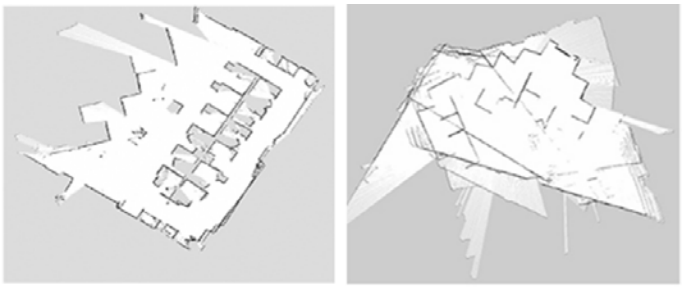
A. 2D SLAM

The ability to build a map of the unknown environment and localize itself, named as SLAM, is one of the most important abilities for robots to operate fully autonomously in USAR scenarios. Most existing 2D SLAM algorithms are based on probabilistic representations. The advantage is the robustness to measurement noises and the capability to formally represent uncertainty in the measurement and estimation process. Furthermore, most probabilistic SLAM algorithms are built upon the Bayes rule. HectorSLAM [5] and Gmapping [6] are two typical Bayes based methods, and open source implementations are available as ROS packages.

HectorSLAM is a 2D SLAM system based on robust scan matching [5]. This module focused on the real-time estimation of the robot movement, making use of LiDARs with high update rate and low measurement noise. The odometry information is not required, which gives the possibility to implement this approach in tracked robots traveling in uneven terrains. The 2D pose estimation is based on the optimization of the alignment of beam endpoints with the map obtained so far. The endpoints are projected in the actual map where the occupancy probabilities are estimated. Scan matching is solved using a Gaussian-Newton approach, which can obtain the rigid transformation that best fits the laser beams with the map. In addition, a multi-resolution map representation is used to avoid getting stuck in the local minima.

Gmapping is a widely used SLAM package [6] in ROS. This algorithm is based on Rao-Blackwellized Particle Filter (PF) . The algorithms based on PF generally requires a large number of particles to achieve accurate estimation results, which increases the computational load. It combines odometry and scan matching in order to reduce the number of particles.

However, in the rescue environment, the odometry is unreliable, which makes Gmapping not suitable. Different from Gmapping, HectorSLAM relies only on scan matching, which is an advantage in USAR tasks. Fig. 4(a) and Fig. 4 (b) show the SLAM results of HectorSLAM in the office environment and the rescue environment, respectively. The mapping result in Fig. 4(b) contains distortions, , which is because the rescue environment is unstructured. In the next subsection, we will introduce how to deal with this problem.



(a) Office environment (b) Rescue environment

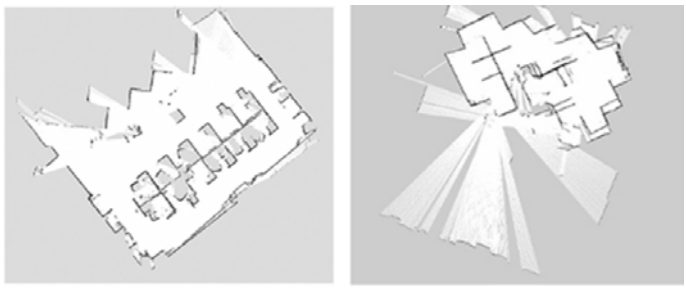
Fig. 4 The SLAM results of HectorSLAM in the office environment with even terrains and the rescue environment with uneven terrains.

B. *Active LiDAR Adjustment*

Robots for USAR are usually used in unstructured environments with uneven terrains. Therefore the sensor data might be spurious if the sensors are rigidly coupled to the robot. The challenge of uneven terrains has been added into the RoboCup rescue competition by using 10° and 15° pitch/roll ramps since 2007 [4].

One way to overcome these problems is to use a 3D LiDAR instead of a 2D LiDAR. The registration of the 3D scans would build a exact global map, and the pose of the robot can also be calculated. However, it spends much more time to do the 3D scan registration with the global map than the scanning and matching in 2D.

As a compromise, we have designed a cheap stabilizer with two servos to adjust the orientation of the 2D LiDAR, based on the readings from the MTI sensor As a result, the 2D LiDAR can be kept on the horizontal plane even when the robot is traversing on uneven terrains. The rotation/tilt stabilizer unit mounted on the robot is shown in Fig. 2. Fig. 5 shows the result of HectorSLAM with LiDAR Stabilizer.



(a) Office environment (b) Rescue environment

Fig. 5 The SLAM results of HectorSLAM with LiDAR Stabilizer in the office environment with even terrains and the rescue environment with uneven terrains.

IV. Autonomous Exploration

A fully autonomous USAR robot must explore the rescue environment and search victims autonomously. This problem can be separated into three questions:

1) Selecting a target point: Where should the robot go next?
2) Planning a path: Which way should the robot take to go to the target?
3) Computing the control command: What action should the robot do?

A. *Frontier-based Exploration*

The primary problem of autonomous exploration is: based on existing knowledge about the real world, where should the robot move to efficiently acquire new information?

Yamauchi proposed a frontier-based approach in [7] to determine the next exploration target. The approach uses the occupancy grid. When a grid map has been built, all the grid can be divided into three categories: Free, Unknown and Occupied. The primary idea is as follows: in order to get new information, going to a frontier which separates known regions from unknown regions. The frontier here is a cell in the occupancy grid which is marked as free but has a neighboring unknown cell (Fig. 6) . A segment of adjacent frontiers is considered as a potential target if it is large enough for the robot getting though. If more than one potential target are detected, the closest one is selected. Fig. 7 shows the result of extracted frontiers.

A disadvantage of directly extracting frontier from the original map is that the extracted frontier may be close to obstacles. To overcome this problem, the Inflated Obstacle method has been utilized. This method will transform the cells within a certain distance to obstacles as Occupied, thus the frontiers will not be extracted in these areas. Fig. 8 shows the

result of extracted frontiers in the occupancy grid with inflated obstacles.

Fig. 6 Frontier detection. (a) The green grid cell has a neighboring Unknown cell, so it is a frontier; (b) The green grid cell has no any neighboring Unknown cell, so it is not a frontier.

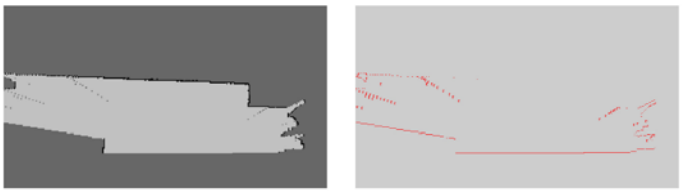


Fig. 7 Occupancy grid and extracted frontier. (a) Black area is Occupied, white area is Free and gray area is Unknown;. (b) Red point are detected as frontiers.

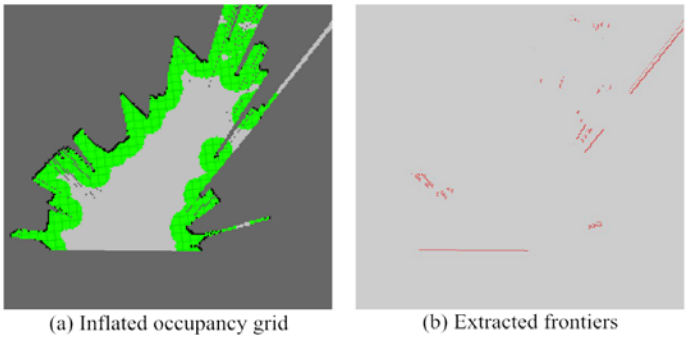


Fig. 8 Occupancy grid with inflated obstacles and the extracted frontier. In (a), green areas are the inflated obstacles.

Unknown areas could be occupied or free areas, which have not been known by robots. In the narrow rescue environment, the probability of the unknown areas being obstacles is quite high. Furthermore, the computational complexity of path planning will increase as the scale of the built map, so the frequency of path planning will not be high for autonomous UASR robots. If the robot takes one frontier as the target point, and the frequency of path planning is low, or the robot's moving distance to the target is small, the robot may crash with unknown obstacles.

Fig. 9 shows such a example where the robot may be stuck. The red point represents the robot's position, and the black curve represents the planned path. In this situation, because of the low frequency of path planning, the robot will not re-plan the path before arriving the target point, so the robot may crash with obstacles. The worst result is that the robot can not move any more.

To deal with this problem, we proposed a method named as Fake Inflated Obstacle, where those extracted frontiers located between free areas and unknown areas are considered as obstacles, then these obstacles are inflated, and finally new frontiers can be searched between these inflated areas and free areas. Using this method, the distances between the target point and the actual obstacles are enough for our robot's safe movement. The extracted frontiers from fake inflated obstacles are shown in Fig 10.

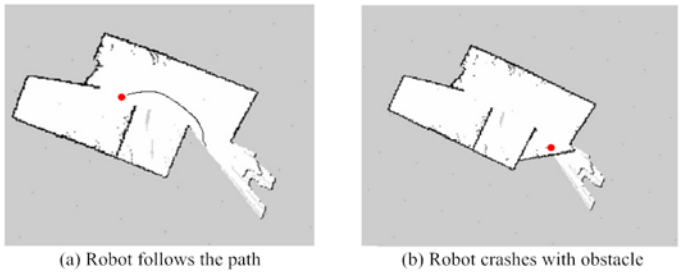


Fig. 9 A example of the robot being stuck

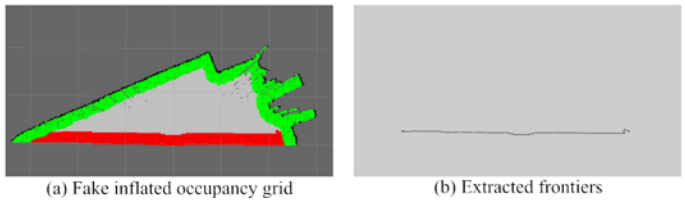


Fig. 10 Occupancy grid with fake inflated obstacles and the extracted frontier.

B. Planning the Optimal Path

When a target has been selected by the frontier-based method, the problem turns into optimal path planning. Javis and Byrne [8] proposed the distance transform to find the closest way from an arbitrary starting point to a fixed target. The distance transform of an occupancy grid calculates the cost to reach the target cell for each free cell. The cost between two cells (without obstacles between them) can be the chessboard distance, city block distance, or the Euclidian distance. After the distance transform has been applied for each cell of the grid, the shortest path from any cell to the target cell can be searched simply by following the steepest gradient. The chessboard distance d of a cell $c_{i,j}$ to target cell $c_{a,b}$ is defined as follows:

$$d = \left|\left(c_{a,b}(x) - c_{i,j}(x)\right)\right| + \left|\left(c_{a,b}(y) - c_{i,j}(y)\right)\right| \quad (1)$$

The distance transform always chooses the shortest way between two cells. But this way is not always desirable for robots, because it may lead the robot crash against the wall or go through a narrow passage. In [9] and [10], a security component has been added to the distance transform, i.e., an obstacle transform which calculates the distance to the closest obstacle for each cell. The path transform Φ of a cell c to reach the target cell c_g is then defined as follows:

$$\Phi(c, c_g) = \min_{C \in \chi_c^{c_g}} \left(l(C) + \alpha \sum_{c_i \in C} c_{danger}(c_i) \right) \quad (2)$$

with $C \in \chi_c^{c_g}$ is the set of all possible paths from c to c_g, l(C) the length of the path C, $c_{danger}(ci)$ the cost function for the

"discomfort" of entering cell c_i, and α a weighting factor ≥ 0. The length l(C) of the path C can be calculated incrementally:

$$l(C) = l(c_0, \cdots, c_n) = \sum_{i=0}^{n-1} d(c_i, c_{i+1}) \quad (3)$$

where d denotes the distance between two cells (e. g. the chessboard distance).

The "discomfort" cost c_{danger} of a cell is calculated based on the obstacle transform of this cell. For distances to the wall that are smaller than half of the size of the robot, the cost should be very high. Zelinsky's choice for such a cost function is given in (4).

$$c_{danger}(c_i) = \begin{cases} \left((X - \Omega(c_i))\right)^3, & \text{if } \Omega(c_i) <= X \\ 0, & \text{else} \end{cases} \quad (4)$$

The constant X determines the minimum distance to obstacles, which depends on the size of the robot, the accuracy of the sensors, and the map. The α in (2) determines how far the path keeps away from obstacles.

C. Controller

Based on the SLAM and exploration planning algorithms mentioned above, the robot can build the 2D grid map and plan a path to the next frontier. Ideally, the robot can explore the environment autonomously after integrating a simple controller to compute the commands to drive itself. However, different from virtual simulation environment or ideally indoor scenarios, real disaster sites are filled with unstructured terrains. Coupled with the inaccuracy of tracked vehicles, it is quite challenging to realize accurate control of the robot to follow the exploration path. A simple controller, which directly produces the velocity command by calculating the biases of current position and orientation of the robot with the target, can not perform well in real world experiments. Therefore, we propose a novel controller combining the exploration planer and multi-sensor information to overcome the influence of challenging terrains. The inputs of the controller are LiDAR data, IMU data, the current position and orientation of the robot and the target.

Firstly, we propose a piece-wise method to evaluate whether the robot has reached the target point:

1) If the robot is exactly close to the target point, the target is reached. The condition of target reached is as follow:

$$\Delta x^2 + \Delta y^2 \leq {d_{\min}}^2;$$

2) If the robot is within a certain distance to the target point, and the robot's current orientation is within a certain angle of the target orientation, the goal is reached. The condition of target reached is as follow:

$$\Delta x^2 + \Delta y^2 \leq d^2 \ and \ \Delta\alpha \leq \theta$$

where Δx and Δy denote the biases of position between the robot and the target. d_{min} and d are the tolerances of translation, and $d_{\min} < d$. $\Delta\alpha$ is the bias of angle between the robot and the target and θ is the tolerance of orientation.

This kind of evaluation can make the robot avoid wasting time to move to very close target points. The robot's velocity commands can be generated by computing the bias of the robot pose and the target point.

Secondly, the point cloud data acquired by the LiDAR are used to compute the "acting force" of the obstacles to the robot. The ideal case is that the robot can keep such a distance to the obstacles that the robot will not crash with the obstacles, and at the same time, the robot can detect and recognize the landmark information located on the obstacles. Therefore, the "acting force" of each point to the robot is computed according to the following principle:

1) the robot is encouraged to stay away from obstacles at a distance of d_{opt} and

2) the robot will not get closer than d_{min}.

The acting force is then computed as follows:

$$f(p_i) = \begin{cases} -\infty & \text{if } d \leq d_{\min} \\ (d - d_{opt})^3 & \text{if } d_{\min} < d \leq d_{\max} \\ F_{\max} & \text{if } d > d_{\max} \end{cases} \quad (5)$$

This force is used to compensate the robot's velocity commands in the controller, so the robot will be kept at a certain distance to the obstacles. Then the acquired IMU information such as the robot's roll and pitch angles can be used to evaluate the terrains. According to the robot's roll and pitch angles, we design the following strategy to improve the robustness in traversing the uneven terrains:

1) When the robot is on the uphill terrain, the robot's linear velocity commands are added by $v_x' = v_x / \cos(\alpha)$ to make sure that the robot can traverse the uphill terrain successfully. Where v_x' and v_x denote the forward direction linear velocity command, and α is the angle of pitch.
2) When the robot is turning on the tilted terrain, the robot's angular velocity commands are added by $\omega_z' = \omega_z / \cos(\beta)$ to make sure that the robot can make a turn on the tilted ramp reliably. Where ω_z' and ω_z denote the angular velocity command, and β is the angle of roll.
3) When the angle of roll or pitch is too large, which means that the robot may turnover, the robot should move to the opposite direction.

The experimental results show that using this kind of multi-sensor based controller, the exploration efficiency and robustness can be improved, and the robot can explore the full rescue environment of the 2016 RoboCup China Open RRL competition, as shown in the accompanying video.

V. Victim Detection and Localization

Reliable detection of human victims in unstructured post-disaster environments is a key issue for USAR robots, which is challenging in the rescue environment with low illumina-

tion, dust and smoke when using normal visible light camera. Therefore we use a thermal camera (Optris P640) to recognize the simulated victims autonomously using a blob detection algorithm. After segmenting the thermal image with a threshold like 36°, the connected warm regions can be regarded as victims.

After detecting the victim successfully, the position of the victim should be estimated on the built map. The image coordinate of the victim can be used to evaluate the victim's direction in the camera coordinate system. Because the victim should be located on the obstacles, using the victim's direction and the camera's orientation, the victim's position can be estimated by searching the nearest obstacle on the built map along the victim's direction.

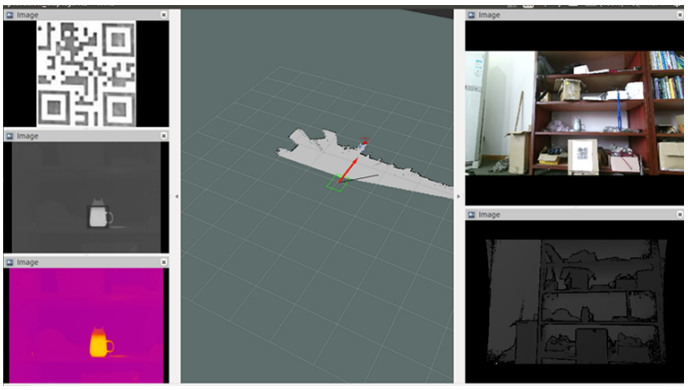


Fig. 11 The result of victim detection and localization, where the found victim is marked by red dot in the map.

VI. Experimental Results

The RoboCup RRL competition provides a systematic benchmark for testing and evaluating teleoperated and autonomous USAR robots in a simulated post-disaster environment. We present the results achieved when participating the RoboCup China Open 2016. In this competition we won the championship based on exploring the rescue environment autonomously.

Figure 12 shows the map of explored arena and the marking of the found victims and recognized QR-codes on the map. It should be noted that the robot successfully found 5 victims autonomously and the total number of victims was 8 in the final competition. The number of the recognized QR-codes correlates with the fraction of the arena that was explored by the robot during the competition. Videos of the final competition and experiments are available on our website:

http://www.trustie.net/organizations/23/videos

VII. Conclusion

Autonomous capabilities of rescue robots in USAR tasks offer an exciting prospect of rescue operations. This paper describes the approaches and technical achievements in designing the fully autonomous USAR robot system of the Team NuBot, which won the 2016 RoboCup China Open RRL championship.

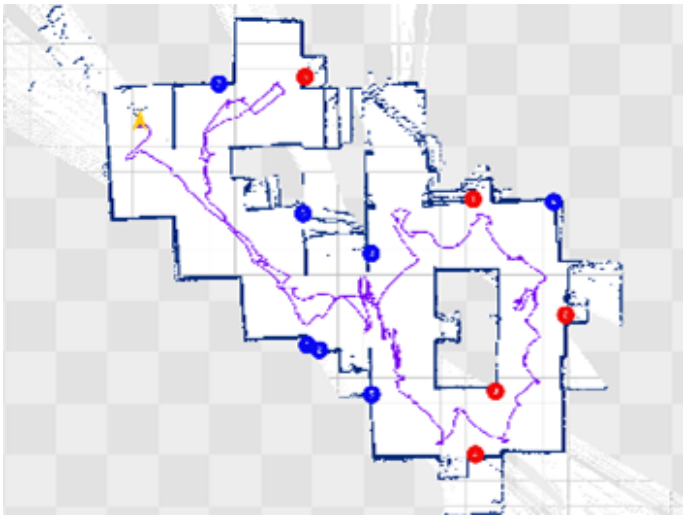


Fig. 12 The autonomous exploration results of our rescue robot participating the 2016 RoboCup China Open RRL. The starting pose of the robot is marked by yellow arrows, the detected victims marked by red dot, and the recognized QR-codes marked by blue dot. In the final competition the robot discovered 5 victims correctly and autonomously.

Acknowledgment

Our work is supported by National Science Foundation of China (No. 61403409 and No. 61503401) and the graduate school of National University of Defense Technology.

References

[1] Sheh, Raymond, et al. Advancing the State of Urban Search and Rescue Robotics Through the RoboCupRescue Robot League Competition. Field and Service Robotics. Springer Berlin Heidelberg, 2014: 127-142.

[2] DEMSKI P, MIKULSKI M, KOTERAS R. Characterization of Hokuyo UTM-30LX laser range finder for an autonomous mobile robot. Advanced Technologies for Intelligent Systems of National Border Security. Springer. 2013: 143-53.

[3] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA Workshop on Open Source Software 2009.

[4] Pellenz, Johannes, and D. G. D. Paulus. "Robbie: A Fully Autonomous Robot for RoboCupRescue." Advanced Robotics, 23(9): 1159-1177, 2009.

[5] Kohlbrecher, Stefan, et al. "A flexible and scalable SLAM system with full 3D motion estimation." Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on IEEE, 2011: 155-160.

[6] Grisetti, G., C. Stachniss, and W. Burgard. "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters." Robotics IEEE Transactions on, 23(1): 34-46, 2007.

[7] Yamauchi, B. "A frontier-based approach for autonomous exploration." IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997. Cira'97. Proceedings IEEE, 1997: 146-151.

[8] R. A. Jarvis and J. C. Byrne. Robot navigation: Touching, seeing an knowing. In Proc. Australian Conf. on Artificial Intelligence, Melbourne, Australia, 1986.

[9] Alexander Zelinsky. Robot navigation with learning. Australian Computer Journal, 20(2): 85-93, 1988.

[10] Alexander Zelinsky. Environment Exploration and Path Planning Algorithms for a Mobile Robot using Sonar. PhD thesis, Wollongong University, Australia, 1991.